UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Planejamento da Expansão de Sistemas de Transmissão assistido por Sistemas Baseados no Conhecimento

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA

PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA

HERCÍDIO DE PAULA SILVEIRA NETO

FLORIANÓPOLIS, DEC-1994

PLANEJAMENTO DA EXPANSÃO DE SISTEMAS DE TRANSMISSÃO ASSISTIDO POR SISTEMAS BASEADOS NO CONHECIMENTO

HERCÍDIO DE PAULA SILVEIRA NETO

ESTA DISSERTAÇÃO FOI JULGADA ~~ADEQUADA~~ PARA OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA ELÉTRICA
PARA A ÁREA DE PLANEJAMENTO DE SISTEMAS DE ENERGIA ELÉTRICA, E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO DA UNIVERSIDADE FEDERAL DE SANTA CATARINA

Prof. Marciano Morozowski Filho - M.Sc.
Orientador

Enio Valmor Kassick - D.Eng.
Coordenador do Curso de Pós-Graduação em Engenharia Elétrica

BANCA EXAMINADORA:

Prof. Marciano Morozowski Filho - M.Sc.
Presidente - Orientador

Prof. Ildemar Cassana Decker - D.Sc.

Prof. Hans Helmut Zürn - Ph.D.

Prof. Paulo Freitas - M.Sc.

RESUMO

Em vista das dificuldades inerentes ao processo de planejamento da transmissão, vem sendo desenvolvidos métodos ditos de *síntese,* que procuram identificar os reforços que atendem com maior eficiência (relação custo/benefício) a uma determinada configuração de geração e carga. Sob esta ótica, foram desenvolvidos no Brasil dois modelos de síntese de sistemas de transmissão: o **TANIA** e o **SINTRA**. Tais modelos são vistos como *caixas pretas*, pois o usuário não tem como incluir novas heurísticas ou novos critérios para otimização, sem alterar o código fonte das subrotinas que os compõem.

Com os presentes paradigmas de programação orientada a objeto e sistemas baseados no conhecimento (*knowledge based systems*), o planejador pode elaborar protótipos ditos *inteligentes* que visam assistir o usuário no processo decisório. Nesse contexto, o presente trabalho visa descrever um protótipo para síntese da transmissão, denonimado **SAPIENS** (*Sistema Adaptativo e Interativo de Planejamento da Expansão*), que utiliza técnicas e conceitos desenvolvidos originalmente na área de inteligência artificial. Este protótipo se caracteriza por ser um modelo híbrido, contendo três módulos básicos: o *motor de simulação*, responsável pelo processamento numérico; a *base de conhecimentos*, incorporando as *regras de produção* (heurísticas de planejamento), a *base de fatos* (dados sobre a topologia da rede e parâmetros de seus circuitos) e os *métodos* (paradigma OOP); o *motor de inferência*, responsável pelo processamento lógico. Esta estrutura modular lhe confere uma singular flexibilidade, permitindo que se adapte às necessidades de seus usuários. A inclusão de novas heurísticas e a eventual reformulação dos algoritmos de simulação empregados, ou mesmo, a escolha de um novo motor de simulação, pode ser feita à parte das demais rotinas que compõem o sistema. Este protótipo contém também uma interface gráfica que permite visualizar o estado de síntese, através de um diagrama unifilar da rede de transmissão em estudo.

A metodologia proposta visa não só liberar o usuário de pensar em fatos ditos *triviais* (comuns na prática), mas também armazenar o conhecimento de especialistas em planejamento, auxiliando no processo de treinamento das novas gerações de engenheiros que os seguirão.

ABSTRACT

The transmission system planning is a highly complex and specialized subproblem of the overall power system planning problem. The inherent difficulties of this problem led to the development of the so called "transmission synthesis" models, in contrast with the "pure" transmission system analysis models traditionally used in power system planning (load flow, stability, etc.).

Two models of this kind have been developed in Brazil: TANIA and SINTRA. TANIA is a dynamic optimization model, whereas SINTRA is an interactive, static, heuristic based synthesis model.

Both programs are "black boxes", in the sense that modification of existing rules or addition of new planning rules require extensive and detailed recodification of the original code. This is a painstaking and risky activity, as far as we can never be sure that all the loops were visited.

Knowledge based systems and object oriented programming paradigms provide a new and flexible framework for prototyping new applications, one of which is the object of this dissertation. The SAPIENS model is a decision support, knowledge based, system adequate for transmission system planning.

SAPIENS was built upon concepts and techniques originally developed in the Artificial Intelligence discipline. This prototype is a hybrid model comprising three modules: the simulation motor, that responds for the numeric processing tasks, the knowledge base, that contains the production rules (planning heuristics), the facts base, containing the network and component data, and the methods (OOP paradigm), as well as the inference motor, which makes all logical processing tasks.

This structure gives the prototype modeling flexibility, making it easy to modify planning rules, in order to fit the model to the most diverse planning environments. The inclusion of new

heuristics and the addition of new simulation motors may be easily done, since these modifications do not require modification in the remaing parts of the system. The prototype graphical interface allows easy visualization of thenetwork state at each step of the synthesis process.

This model and the underlying methodology are useful not only to relieve the planner from thinking in trivial facts but also to register specialized knowledge and as a training tool able to prepare new generations of engineers.

NOTAÇÃO E SIMBOLOGIA

Os * (asteriscos) se referem a notas de rodapé sobre um tópico em especial.

Os números entre [00] (colchetes) atuam como referência de publicações científicas. Tais referências estão contidas no último capítulo desta dissertação.

Palavras escritas em itálico, bem como termos entre aspas, indicam termos técnicos assumidos como no original, bem como metáforas aceitas em consenso por especialistas. São empregadas também para citar expressões coloquiais.

A notação e simbologia da modelagem de fluxo de potência linearizado está presente no Apêndice A, responsável por sua definição e descrição.

Non Nobis Domine,

Non Nobis,

Sed Nomine Tuo, da Gloriam

(Ordem dos Cavaleiros Templários da Idade Média)

I understood how gravity behaved,

but not how it worked.

(Sir Isaac Newton)

À minha mãe, que com amor e compreensão pôde suportar minhas dificuldades.

A Deus, o princípio maior que norteia as criações do Universo.

AGRADECIMENTOS

Ao Prof. Marciano Morozowski Filho, pelo empenho, desprendimento e apoio dados como orientador desta monografia e como especialista consultado, fornecendo-me importantes subsídios no trato da questão do Planejamento de Sistemas de Potência.

Aos senhores César Ghisi Silvestre, Luiz Gastão C. Souza, especialistas da ELETROSUL (Centrais Elétricas do Sul do Brasil S.A.) em Planejamento de Sistemas de Potência, pelas entrevistas concedidas. Aos senhores Gerson Couto de Oliveira e Silvio Binato, especialistas do CEPEL (Centro de Pesquisas da ELETROBRÁS), por nos conceder o código fonte do modelo SINTRA, versão de Setembro de 1992, utilizado como motor de simulação do protótipo implementado.

Ao Prof. Paulo Freitas, por nos dar acesso ao shell KAPPA-PC, da Intellicorp Inc.

Aos demais professores do Departamento de Engenharia Elétrica, pelo embasamento teórico e troca de experiências.

Aos amigos Fabíola Sena Vieira, Gustavo Morozowski e Nixon Savaris, bolsistas de iniciação científica, pelo apoio moral e técnico no desenvolvimento do protótipo implementado, descrito a seguir. Aos colegas de curso e demais amigos, membros da fraternidade universal da qual somos parte integrante.

Ao Conselho Nacional de Pesquisa e Desenvolvimento Tecnológico - CNPQ, pelo apoio financeiro.

À Universidade Federal de Santa Catarina pelo acervo técnico e bibliográfico.

SUMÁRIO

Página

LISTA DE QUADROS

Página

LISTA DE FIGURAS

Página

CAPÍTULO 1 - INTRODUÇÃO

1.1. CONSIDERAÇÕES GERAIS

Antes da descrição dos objetivos deste trabalho, algumas definições serão apresentadas com o objetivo de esclarecer o *processo*[*] de planejamento de sistemas de potência.

Como consta em [52], o objetivo básico do planejamento de sistemas de potência pode ser definido como a determinação da data de entrada em operação dos equipamentos e instalações de geração, transporte e distribuição de energia elétrica, considerados necessários para atender o crescimento do mercado de energia elétrica. Em geral, o processo de planejamento repousa sobre uma solução de compromisso entre o nível de confiabilidade desejado e os custos de investimento e construção relacionados. Este compromisso pode ser melhor compreendido pela Figura 1.1.

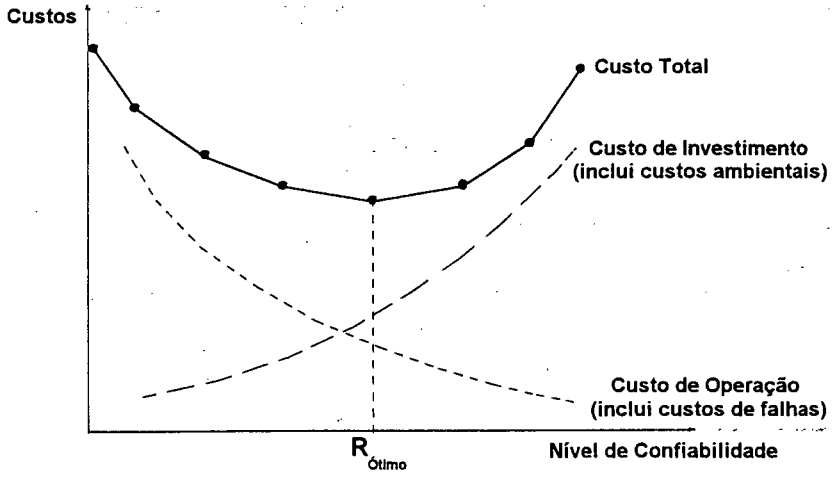


FIGURA 1.1 - PLANEJAMENTO BASEADO NA CONFIABILIDADE E CUSTOS

[*] O conceito de processo decorre da natureza sistêmica da atividade de planejar.

Assim colocado, o problema de planejamento pode ser enunciado como um problema de otimização multiperíodo que, devido às características dos sistemas de potência, resulta ser de grande porte, não-linear e estocástico.

Como consequência da complexidade do problema de planejamento e considerando que as incertezas nas projeções de mercado, de custos de combustível e das tecnologias se ampliam na medida em que se estende o horizonte de planejamento, surge a necessidade de decompor o problema de expansão em subproblemas de menor porte e complexidade, passíveis de solução com as metodologias e recursos computacionais disponíveis.

Assim, é usual, nas empresas de energia elétrica, realizar os estudos de planejamento em etapas, de acordo com critérios de decomposição temporais, espaciais e funcionais, conforme indicado no Quadro 1.1.

QUADRO 1.1 - DECOMPOSIÇÃO DO PROCESSO DE PLANEJAMENTO

CRITÉRIO	ETAPAS
Temporal	• Longo prazo (15 a 30 anos) • Médio prazo (5 a 15 anos) • Curto prazo (3 a 5 anos)
Espacial	• Nível nacional • Nível regional • Nível local
Funcional	• Geração (produção) • Transmissão (transporte) • Distribuição (atendimento ao mercado consumidor)

A decomposição do problema global, ao mesmo tempo que viabiliza a solução do problema de planejamento, requer o adequado encadeamento dos resultados de cada etapa, de forma a garantir a consistência do plano como um todo. Este encadeamento é usualmente realizado através do estabelecimento de um fluxo de informações, conforme ilustrado na Figura 1.2, que especifica os dados de entrada e os principais produtos de cada etapa. Os tipos de estudo realizados em cada etapa do planejamento são detalhados no Quadro 1.2.

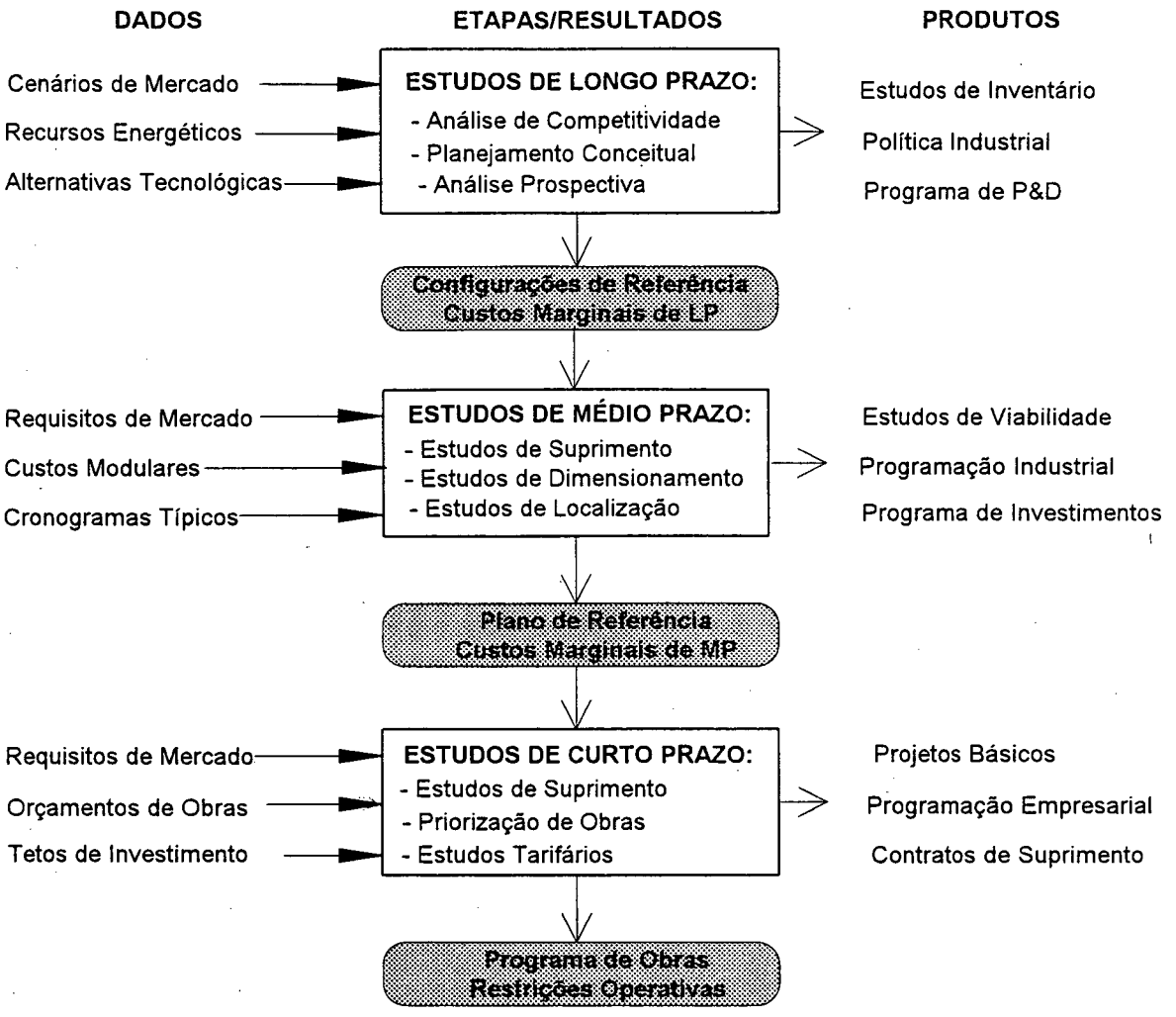


FIGURA 1.2 - FLUXO DE INFORMAÇÕES NO PLANEJAMENTO

A decomposição apresentada, ao mesmo tempo que simplifica a resolução do complexo problema de planejamento da expansão dos sistemas elétricos, cria a necessidade de mecanismos de coordenação das etapas, de modo que o produto final, o Programa de Obras, seja consistente. Enfatiza-se que a consistência deve ser avaliada até o nível de concepção e de dimensionamento dos projetos, onde efetivamente se evidencia a qualidade do planejamento.

QUADRO 1.2 - TIPOS DE ESTUDOS REALIZADOS NO PLANEJAMENTO DA EXPANSÃO

HORIZONTE	TIPO DE ESTUDO	SUB-SISTEMAS/COMPONENTE
Longo prazo	• Estudos de Inventário • Avaliações Tecnológicas • Planejamento Conceitual	• Geração • Geração/Transmissão • Geração/Interligações
Médio prazo	• Expansão • Dimensionamento • Localização	• Geração/Transmissão • Usinas/Circuitos • Usinas Térmicas /SE's
Curto prazo	• Suprimento Energético • Priorização de Obras • Limites de Equipamentos	• Geração/Transmissão • Transmissão/Subtransmissão • Reatores/TF's/Disjuntores

Em geral, os planos de expansão com horizonte de estudo a curto e médio prazo podem ser efetivados com apoio de modelos de análise tradicionais, tais como: programas de fluxo de potência, estabilidade transitória e curto circuito. Entretanto, os estudos a longo prazo são mais complexos, sendo influenciados pelos fatores a seguir [43]:

a) natureza combinatória do problema de planejamento devido à presença de variáveis discretas de decisão;

b) grande número de variáveis e restrições envolvidas em função do porte dos sistemas tratados;

c) incertezas inerentes aos custos dos novos equipamentos, aos planos de geração e às previsões de demanda, bem como às características de operação destes equipamentos;

d) caráter dinâmico na tomada de decisões ao longo dos vários períodos de estudo;

e) não-linearidade das grandezas envolvidas no problema.

Em decorrência destes fatores, e visando aumentar a eficiência do trabalho de planejamento, foram desenvolvidas algumas ferramentas computacionais baseadas em métodos heurísticos, como mostra a Figura 1.3 ([1], [22], [61] e [75]).

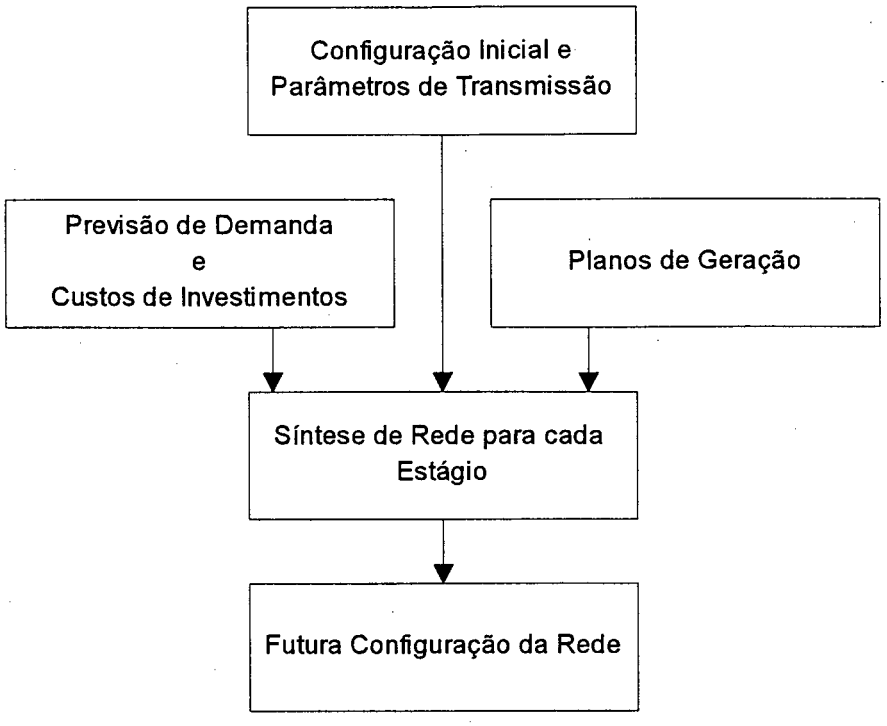


FIGURA 1.3 - PROCESSO DE SÍNTESE DE UMA REDE DE TRANSMISSÃO

Não se pode dizer que seja possível criar um modelo computacional que generalize o processo de planejamento. Contudo, é possível desenvolver modelos aproximados, simplificados, que gerem soluções satisfatórias a partir de heurísticas de planejamento propostas por especialistas da área ([6], [7], [23], [28] e [31]).

A inclusão de heurísticas em modelos computacionais é uma forma artificial de se aproximar do modelo mental humano, facilitando a integração destes programas com seus usuários. Este é o objetivo da Inteligência Artificial (IA), uma área de pesquisa cujo avanço conseguido ao longo dos últimos anos, em vários setores de ciência aplicada (ciências exatas, humanas e biológicas) serve como prova do interesse que existe por parte de pesquisadores ligados ao assunto ([31], [39], [40], [54], [60], [70], [72] e [73]). Devido à evolução das técnicas de IA, novos paradigmas de programação estão surgindo, sendo abordado com mais detalhes nesta dissertação o de *Sistemas Baseados no Conhecimento* (SBC) e programação orientada a objeto (OOP).

Na área de Operação de Sistemas Elétricos, já existem alguns SBC's, propriamente ditos, voltados para o diagnóstico e controle da rede de energia elétrica em tempo real, atuando como *filtros* para alarmes do sistema de proteção* ([4], [60] e [74]).

Na área de planejamento de sistemas de potência, aplicações atuais de SBC's incluem: treinamento; análise da consistência dos dados de entrada; estudos de estabilidade; previsão de demanda; esquema de desligamentos de usinas nucleares; etc. [4].

Como consta em *Monticelli* [46]:

> *"...A utilização do computador digital e dos métodos de síntese não substitui, mas altera a qualidade do trabalho do planejador, que passa a ser capaz de gerar um número maior de estratégias de expansão, reduzindo ao mesmo tempo o fator subjetivo na escolha de alternativas."*

1.2. OBJETIVO DO TRABALHO

A presente dissertação objetiva desenvolver uma metodologia mais flexível para modelagem do processo de planejamento de sistemas de potência. Tal metodologia é útil no desenvolvimento de sistemas experimentais, graças ao emprego dos paradigmas de SBC's que, por modularizar o projeto global, tornam estes protótipos mais flexíveis do que os sistemas comerciais disponíveis. Como exemplo de aplicação da técnica proposta, descreve-se o protótipo SAPIENS (*Sistema Adaptativo e Interativo de Planejamento da Expansão*), um SBC que dá apoio ao processo de síntese estática no âmbito do planejamento da expansão de redes de transmissão. Este protótipo,

---

* Como exemplo, pode-se citar o programa CRAFT (*Customer Restoration and Fault Testing Expert System*), um SBC que auxilia na recomposição do sistema elétrico após a ocorrência de uma falta no sistema de transmissão. Além deste, existem outros aplicativos a citar, voltados ao controle do perfil de tensão pelo despacho de reativos, análise dinâmica de segurança, colapso de tensão, fluxo de potência ótimo, etc.

implementado como um sistema híbrido, agrega as rotinas numéricas do modelo SINTRA [7] com heurísticas de planejamento adquiridas mediante entrevistas com especialistas da área.

As facilidades abertas ao operar com ambiente *Windows* (ambiente gráfico multitarefa), aliadas à representação do conhecimento (modelagem) sob a forma de *frames*, juntamente com o uso de paradigmas de programação orientada a objeto, permitiu a composição de uma interface homem-máquina bastante amigável no protótipo SAPIENS, da qual faz parte uma interface gráfica que expõe os estágios de síntese, representados na forma de diagramas unifilares de redes de transmissão.

1.3. ORGANIZAÇÃO DA DISSERTAÇÃO

Os capítulos a seguir estão organizados como segue:

Capítulo 2 - Planejamento de Sistemas de Transmissão

Este capítulo apresenta o processo de planejamento em sua forma conceitual, além de apresentar, sucintamente, os atuais modelos de síntese em uso pelas concessionárias de energia elétrica nacionais.

Capítulo 3 - Evolução dos Paradigmas de Programação

Este capítulo apresenta um breve resumo sobre a evolução das técnicas de programação utilizadas na modelagem do processo de planejamento, discorrendo sobre técnicas de Inteligência Artificial, seu surgimento e estado da arte. Por fim, faz citações a sistemas baseados no conhecimento, e programação orientada a objeto, abrindo campo para os próximos capítulos.

Capítulo 4 - Sistemas Baseados no Conhecimento

Este capítulo descreve, de forma conceitual, a arquitetura de sistemas baseados no conhecimento, enfatizando os aspectos de modularidade e flexibilidade presentes nesta forma de programar.

Capítulo 5 - Paradigmas de Programação Orientada a Objeto

Este capítulo expõe os conceitos fundamentais sobre programação orientada a objeto, enfatizando os aspectos de modularidade e flexibilidade de desenvolvimento presentes nesta forma de programar.

Capítulo 6 - Modelagem do Protótipo SAPIENS

Este capítulo propõe uma metodologia para desenvolvimento de SBC's, tema desta dissertação, apresentando o protótipo SAPIENS como um exemplo de implementação. São descritos o motor de inferência, o ambiente de trabalho, a ferramenta de desenvolvimento, a base de conhecimentos (classes e instâncias) e o mecanismo de passagem de mensagens para acionamento do conjunto de métodos encadeados.

Capítulo 7 - Versões Implementadas

Neste capítulo, são apresentadas as últimas versões do protótipo SAPIENS, a fim de mostrar o caráter evolutivo que o mesmo contém, juntamente com o algoritmo de trabalho de cada versão, os quais esboçam a forma como o usuário irá trabalhar com o protótipo.

Capítulo 8 - Estudo de Casos

Neste capítulo, a título de exemplo, é simulado um caso teste com o modelo SAPIENS, com base no qual se discute o desempenho do protótipo, em termos de

tempo de execução e resultados oferecidos, comparando-o com modelos convencionais.

Capítulo 9 - Conclusões e Recomendações

Neste capítulo, são inferidas conclusões sobre o trabalho apresentado e propostas linhas de pesquisa com base no presente estágio de implementação do protótipo.

Apêndice A - SINTRA - Modelagem Conceitual

Este apêndice trata da formulação conceitual do modelo SINTRA e apresenta seu fluxograma operacional.

Apêndice B - Estratégias de Planejamento Adotadas pela ELETROSUL

Este apêndice resume algumas heurísticas adotadas no planejamento de sistemas de transmissão pela ELETROSUL (Centrais Elétricas do Sul do Brasil S.A.), conforme apreendidas em entrevistas com os Engenheiros César Ghisi Silvestre e Luiz Gastão C. Souza.

Apêndice C - SAPIENS - Versão 2.0

Este apêndice apresenta a listagem dos diversos módulos que compõem o protótipo SAPIENS em sua última versão. Este apêndice visa dar subsídios a trabalhos futuros.

Apêndice D - Funções de Biblioteca definidas pelo usuário

Este apêndice, além de mostrar a forma usual de implementar uma DLL (*Dynamic Link Library*) com o auxílio do *shell* KAPPA-PC, apresenta a listagem de três DLL's desenvolvidas e empregadas pelo protótipo SAPIENS em sua última versão. Este apêndice visa dar subsídios a trabalhos futuros.

Referências Bibliográficas

Este capítulo apresenta o acervo bibliográfico referenciado neste trabalho, listado em ordem alfabética do principal autor.

CAPÍTULO 2 - PLANEJAMENTO DE SISTEMAS DE TRANSMISSÃO

2.1. INTRODUÇÃO

O presente capítulo visa discorrer sobre o processo de planejamento a longo prazo da expansão de sistemas de transmissão de energia elétrica. Explanações mais detalhadas podem ser encontradas em trabalhos correlatos ([45] e [46]).

2.2. PLANEJAMENTO DE SISTEMAS DE TRANSMISSÃO

Na prática atual das empresas de energia elétrica, o planejamento se processa passo a passo, com períodos definidos de estudo, através da simulação do desempenho do sistema em regime permanente e, eventualmente, em regime não permanente à frequência nominal (vide Quadro 2.1).

Sob esta ótica, o planejamento de sistemas de transmissão constitui um processo de decisões sequenciais de grande porte, onde a busca de soluções ótimas levou ao desenvolvimento e emprego de técnicas de programação dinâmica e algoritmos de otimização ([15] e [28]). Devido ao alto *custo computacional*[*] envolvido, os esforços se direcionaram na busca de modelos que fossem mais simplificados, mas que apresentassem soluções aceitáveis.

Atualmente, o planejamento é balizado por critérios ditos determinísticos e a comparação de alternativas de expansão se resume ao custo atualizado de cada alternativa, ou seja, admite-se que os *benefícios*[*] são equivalentes.

---

[*] Tempo de execução, memória de trabalho, precisão do microprocessador, etc.

[*] Qualidade e quantidade de energia disponível ao consumo. Os sistemas planejados sob uma ótica determinística atendem a padrões rígidos de qualidade. Assim, pelo menos no que diz respeito à operação em condição normal

QUADRO 2.1 - TAREFAS BÁSICAS NO PROCESSO DE PLANEJAMENTO

ATIVIDADE	OBJETIVO
Coleta e Análise de Dados	Obter, depurar e organizar a estrutura de dados necessária à execução das atividades a seguir.
Análise do Sistema em Regime Permanente	Analisar o sistema em condição normal ou de emergência, para os patamares de carga pesada, intermediária e leve. Abrange: • *Fluxo de Potência em Condições Normais* verifica se o sistema permite o atendimento da demanda, considerando todos os equipamentos em serviço e situação de carga / despacho de geração nominais. Avalia se o carregamento destes equipamentos e as tensões em seus barramentos estão dentro dos limites nominais. • *Fluxo de Potência em Condições de Emergência* analisa as condições operativas em situação de perda de algum componente da rede de transmissão. Detecta as eventuais sobrecargas e os requisitos de suporte de reativo para fazer frente às contingências.
Análise do Sistema em Regime Dinâmico	Avaliar a aptidão da configuração em estudo em manter sua integridade, transitoriamente, quando solicitada por perturbações que envolvam a retirada, temporária ou definitiva, de componentes do sistema (*Estudos de Estabilidade Transitória e Dinâmica*).
Análise de Sobretensões e Curto-Circuitos	Identificar a necessidade de alterações na rede para reduzir o nível de solicitação.
Análise Econômica de Alternativas	Selecionar o plano de expansão mais atrativo entre alternativas tecnicamente viáveis. Para sua execução são necessários os custos unitários de equipamentos, custos modulares para compor os custos globais de SE's, custos indiretos, etc.

A Figura 2.1 mostra um sistema de transmissão hipotético [68]. A rede existente é mostrada em linhas sólidas e as prováveis rotas de expansão em linhas pontilhadas. Como os valores de carga e despacho de geração variam ano a ano, torna-se necessário atualizar a configuração da rede de transmissão (adições de novos circuitos à rede existente) de forma a manter a confiabilidade do sistema na transmissão de energia elétrica desde os geradores até os diversos centros de carga.

---

ou sob contingência simples, os benefícios são considerados idênticos para todas as alternativas consideradas [52].

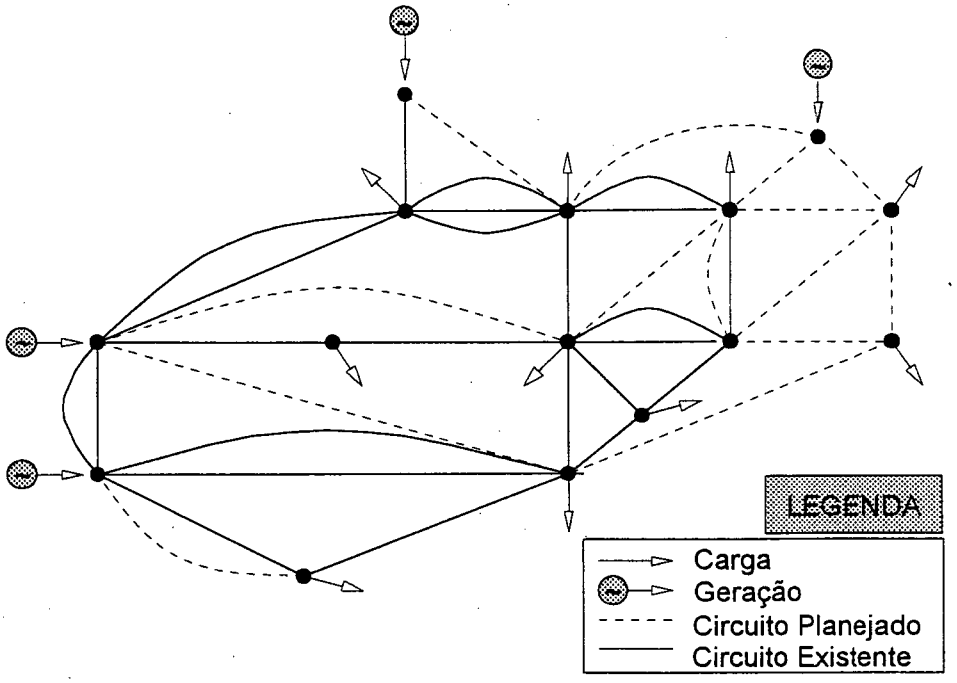


FIGURA 2.1 - ESQUEMA DE UM SISTEMA DE TRANSMISSÃO

Mesmo no caso de sistemas com dimensões reduzidas, como é o caso do esquema apresentado na Figura 2.1, muitas podem ser as configurações de rede que satisfaçam os requisitos de confiabilidade exigidos. Assim, pode-se concluir que a complexidade do processo de planejamento aumenta exponencialmente com o número de configurações de alternativas de expansão de rede de transmissão a analisar.

Para a análise em regime permanente (*planejamento estático*), foco desta dissertação, a ferramenta básica do planejador é o programa de fluxo de potência, desde o modelo não-linear (AC) até os mais simplificados (DC) (vide Apêndice A)*.

* O fluxo de potência linearizado desacoplado rápido estima, com boa precisão e baixo custo computacional, a distribuição de fluxos de potência ativa em redes de potência, como acontece nos modelos iterativos Newton-Raphson, Gauss-Seidel, etc. [18]. Os erros percentuais verificados na utilização do modelo linear são da ordem de ± 5% para os circuitos mais sobrecarregados, valores aceitáveis quando comparados com as incertezas na previsão de carga e nos parâmetros utilizados na expansão a longo prazo do parque gerador. Cabe ressaltar que o modelo DC baseado nas relações P-θ, não apresenta bons resultados para sistemas em média tensão, nos quais os fluxos de potência dependem também das quedas de tensão entre os nós da rede [8].

Observando-se a ocorrência de sobrecargas ou violações de tensão, adotam-se medidas corretivas, que podem incluir:

a) Adição de uma nova linha de transmissão numa faixa de passagem existente, com as seguintes possibilidades:

   i. Circuito simples;

   ii. Circuito duplo com um ou dois circuitos lançados;

   iii. Lançamento do segundo circuito em circuito duplo existente.

b) Seccionamento de linha de transmissão existente (Figura 2.2);

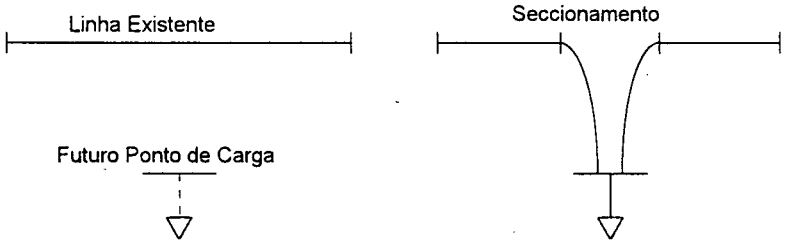


FIGURA 2.2 - SECCIONAMENTO DE LINHAS DE TRANSMISSÃO

c) Mudança de nível de tensão de linha de transmissão existente (Figura 2.3);

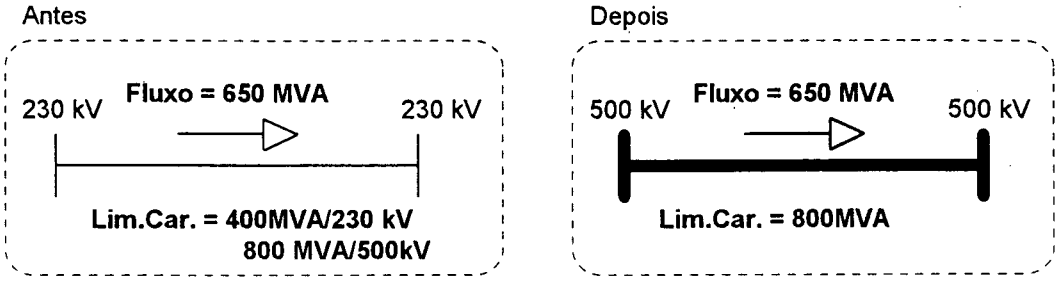


FIGURA 2.3 - MUDANÇA DE NÍVEL DE TENSÃO EM LINHA DE TRANSMISSÃO EXISTENTE

d) Adição de uma nova linha de transmissão em nova faixa de passagem, com as possibilidades antes apontadas;

e) Adição / substituição de transformadores ou mudança de esquemas de chaveamento em subestações (Figura 2.4);

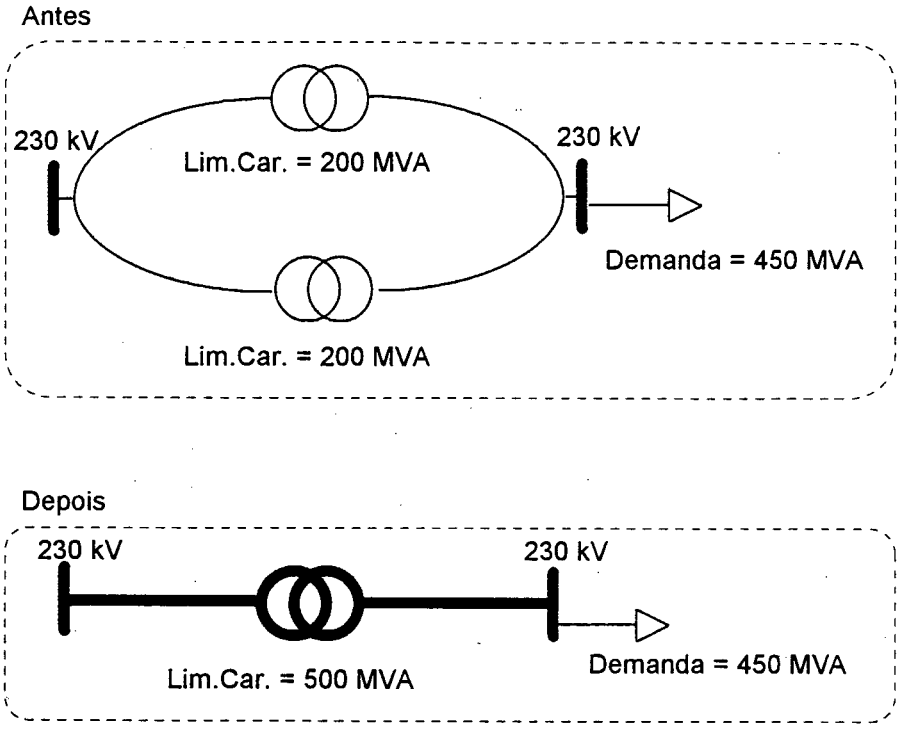


FIGURA 2.4 - SUBSTITUIÇÃO DE TRANSFORMADORES

f) Combinações das várias alternativas descritas acima.

O número de alternativas que se pode escolher é surpreendentemente grande como pode ser demonstrado pelo seguinte exemplo hipotético [67]. A Figura 2.5 mostra seis configurações de linhas de transmissão, quais sejam, um circuito simples e um circuito duplo de baixa tensão, e um circuito simples de alta tensão, todos com e sem compensação série. Uma certa faixa de passagem pode acomodar qualquer uma das seguintes configurações:

a) duas linhas de alta tensão;

b) uma linha de alta tensão e um circuito simples ou duplo de baixa tensão;

c) duas linhas de circuito simples de baixa tensão;

d) duas linhas de circuito duplo de baixa tensão;

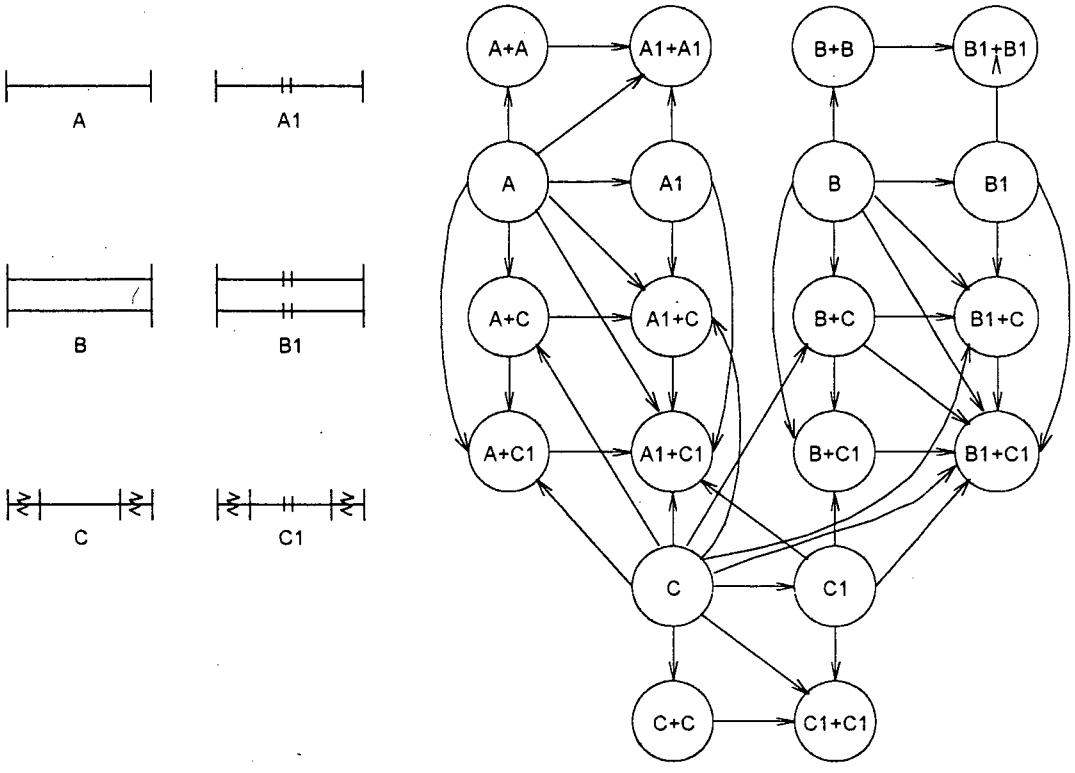


FIGURA 2.5 - NATUREZA COMBINATORIAL DA EXPANSÃO DA TRANSMISSÃO

Além disso, as regras de planejamento excluem a combinação que considera um circuito simples e um circuito duplo de baixa tensão. As regras também especificam que se duas linhas de baixa tensão forem usadas, ambas precisam estar com compensação ou sem compensação série. A Figura 2.5 mostra, sob a forma de um diagrama de transição de estados, as vinte alternativas de transmissão possíveis considerando as restrições assumidas.

Neste exemplo, o número de estágios presentes é vinte. Após cada transição ser efetivada, o número de estágios remanescentes é menor, diminuindo proporcionalmente o esforço computacional no processo de simulação. Em cada caso, a decisão inicial restringe as decisões nos estágios finais.

Configurações mais realistas de sistemas de potência são mais complexas. Por exemplo, estágios adicionais poderão ser gerados ao permitir que um circuito duplo seja incorporado por partes em diferentes anos ou que uma linha de alta tensão opere em baixa tensão durante certo período. Se for permitido remover ou remanejar linhas, outros estados de transição poderão

existir. Situações envolvendo 100 (cem) ou mais estados em uma faixa de passagem são possíveis de ocorrer.

Este exemplo mostra que, mesmo em uma situação simplificada, o número de alternativas em uma faixa de passagem pode ser grande, e que as dimensões do problema crescem à medida que são consideradas simultaneamente outras faixas de passagem disponíveis.

Neste processo de tentativa e erro, todas as decisões quanto ao reforço ou combinações de reforços a adotar cabem ao planejador, que se baseia nas indicações fornecidas pelos instrumentos de análise disponíveis, considerando um conjunto de restrições técnicas e financeiras.

Como o objetivo final do planejamento de sistemas de transmissão é manter a confiabilidade do sistema elétrico em um nível aceitável, é necessário sanar todas violações detectadas pelos processos de análise. Uma vez que violações de carregamento e/ou tensão podem ocorrer em vários pontos do sistema, simultaneamente, raramente um reforço isolado pode resolvê-las no todo.

Em vista dessas dificuldades, inerentes ao processo de planejamento, foram concebidos e vêm sendo desenvolvidos métodos ditos *de síntese*, que procuram realizar de modo eficiente a busca das combinações de reforços mais efetivas em cada caso.

2.3. Evolução dos Modelos de Síntese

A aplicação de computadores digitais ao planejamento e operação de sistemas de energia elétrica teve início na década de 50. Embora algumas propostas de modelos de síntese, baseadas em programação linear, datem de 1960, as atenções de pesquisadores e engenheiros na área de sistemas de potência se concentraram principalmente em métodos de análise como os utilizados no cálculo de fluxo de carga, em estudos de estabilidade, de curto-circuito, etc. Com o passar do tempo, o crescimento das dimensões das redes de transmissão levou novamente à busca por

métodos de síntese que incorporassem a experiência dos planejadores na definição e escolha de alternativas de expansão (*heurísticas de planejamento*). Assim, como consta em [75], modelos de síntese começaram a surgir visando dar respostas a diversas questões relacionadas ao problema de planejamento, tais como:

a) O que será implementado e qual o custo necessário para incorporar um particular esquema de geração a longo prazo?

b) Será que um caminho em particular (sequência de circuitos interligando dois nós, ou barramentos) provê adequada transmissão para um esquema selecionado de geração?

c) Como um sistema de transmissão existente irá afetar o desenvolvimento de um novo sistema de alta tensão?

d) Qual deverá ser a capacidade de transmissão de um caminho selecionado?

e) Qual deveria ser o próximo nível de alta tensão para transmissão e quando se poderia começar a construí-lo?

f) Qual o tipo de transformador, bitola de condutores, etc. a adotar, tendo em vista as necessidades de transmissão?

Historicamente, o desenvolvimento de modelos computacionais para realizar o planejamento a longo prazo de sistemas de transmissão pode ser dividido em três fases distintas [46], como segue:

a) A primeira fase consistiu simplesmente em se automatizar os procedimentos heurísticos utilizados quando os cálculos de simulação de rede eram executados pelo analisador de circuitos ou mesmo manualmente ([1] e [75]).

b) Mais tarde, começaram a aparecer metodologias para substituir os procedimentos heurísticos por métodos de otimização. Dentre os métodos de otimização mais

conhecidos, pode-se citar: programação dinâmica [15], busca em grafo [22], programação linear [32], programação não linear [59] e Branch and Bound [36].

No entanto, poucos desses métodos automáticos apresentaram resultados satisfatórios, levando a concluir que a expansão *ótima* é apenas um aspecto do planejamento com apoio do computador [26].

c) Por fim, o emprego de métodos interativos (*processamento heurístico*), permitiu a participação do planejador durante a obtenção de estratégias de expansão, uma vez que se concluiu ser praticamente impossível, além de geralmente ineficaz, realizar a síntese de redes de transmissão de maneira inteiramente automática ([6] e [7]).

Entre os modelos de síntese desenvolvidos, alguns procuram reproduzir o processo decisório do planejador, constituindo uma categoria de modelos denominados *heurísticos*, ao passo que outros fazem uso de métodos de programação matemática para definir o conjunto de adições mais eficiente em cada situação. Outros ainda procuram combinar os dois processos, com a intervenção ou não do planejador.

Dentre os modelos desenvolvidos no Brasil, pode-se citar o TANIA [28], desenvolvido por Furnas com consultoria da COPPETEC, e o SINTRA (vide Apêndice A), desenvolvido pela ELETROBRÁS e pelo CEPEL com consultoria da UNICAMP.

O primeiro, TANIA, trata de dois níveis distintos de planejamento [28], a saber:

a) expansão a curto prazo, onde se resolve o problema estático de viabilizar a rede para injeções dadas;

b) expansão a longo prazo, onde se concatenam expansões a curto prazo.

O programa TANIA trata do planejamento a curto prazo utilizando o método de otimização Branch and Bound, um algoritmo de busca em profundidade que parte de uma configuração inicial S^0 e examina iterativamente decisões que consistem, cada uma, em adicionar um novo

componente à rede. O problema do planejamento a longo prazo, por sua vez, utiliza um algoritmo de *programação heurística*[*] que visa encontrar um *caminho*[*] de custo mínimo entre um nó inicial e as configurações associadas ao estágio final da expansão.

O TÂNIA é um modelo de síntese automático, no sentido de não prever a intervenção do planejador no processo decisório, e dinâmico, no sentido de procurar definir a sequência mais econômica de decisões ao longo do período de planejamento. Para cada nova adição, ou reforço à rede de transmissão existente, é simulado o comportamento estático do sistema elétrico. Assim, neste modelo, cada alternativa de expansão é analisada individualmente. Entretanto, decai em precisão para redes com cerca de 70 nós [8].

Ele representa uma das primeiras tentativas de automatizar o processo de planejamento ao modelar os critérios de expansão usados pelo setor elétrico brasileiro. Contudo, uma vez que o critério de otimização *depende do planejador* [26], a decisão final sobre *o que* e *quando* planejar, sempre estará nas mãos de um ser humano. Ou seja, o processo decisório envolve necessariamente fatores subjetivos tais como [52]: conjuntura sócio-econômica, apelos por parte de organizações de proteção ambiental, e até mesmo decisões administrativas ou políticas, que visam atender aos ideais de algum grupo de interesse. Assim, modelos deste tipo não permitem interação homem-máquina e se mostram limitados para lidar com o problema do planejamento da expansão a longo prazo de sistemas elétricos de transmissão.

O programa SINTRA (*Síntese Interativa do Planejamento da Transmissão*), por sua vez, é um modelo que permite não só a síntese automática da expansão, mas também a síntese manual. Em sua forma interativa, permite que o planejador execute as tarefas de análise e síntese manualmente, o que contrasta com o modelo TANIA. Esta forma de promover a expansão, facilita o estudo e a elucidação de novas heurísticas de planejamento, baseadas no conhecimento

---

[*] O termo heurística advém do fato de representar o conhecimento prático do engenheiro de planejamento.
[*] O termo *caminho* advém da teoria de grafos.

empírico do planejador, o que melhora substancialmente a eficiência do processo de planejamento.

Como inovação, traz consigo o conceito de *caminho*[*], o que lhe permite analisar mais alternativas de expansão por vez. O critério de avaliação da eficiência de um dado caminho é baseado no cálculo de *índices de desempenho*[*]. O benefício é determinado simulando-se a adição do caminho à rede e avaliando-se a redução da sobrecarga produzida por este. O custo de um caminho é determinado somando-se os custos dos circuitos que compõem o mesmo. Desta forma, o melhor caminho para adição é aquele que possui a menor relação custo/benefício.

Tais modelos, utilizam abordagens baseadas em programação heurística e cálculo de índices de desempenho que permitem selecionar as adições pela ordenação dos melhores candidatos. A inclusão de novas heurísticas de planejamento nos modelos atuais pode melhorar a eficiência dos mesmos. No entanto, a prática tem comprovado que isto é um processo complexo, uma vez que tais modelos são *fechados*, assemelhando-se a uma *caixa preta*. Tal analogia se dá pelo fato do usuário conhecer apenas suas entradas e saídas, nada podendo inferir do seu conteúdo.

Os modelos de síntese, em sua maioria, se baseiam na estimação de índices de desempenho para criação de listas ordenadas de candidatos à adição. O *ranking* dos melhores candidatos possibilita a obtenção de cenários estáticos de expansão, que podem ser usados como ponto de partida para a expansão ótima.

Hoje, a utilização de programas computacionais não substitui, mas altera a qualidade do trabalho do planejador, que passa a ser capaz de gerar um número maior de cenários de expansão, reduzindo ao mesmo tempo o fator subjetivo na escolha de alternativas.

A aplicação de metodologias de Inteligência Artificial a sistemas de potência abre as portas para a implementação de modelos flexíveis. Os programas de simulação existentes fornecem a

---

[*] Conjunto de circuitos com custos associados.
[*] Relação custo/benefício de cada caminho.

base para o processamento numérico. As heurísticas (regras de planejamento) podem ficar em módulos isolados, permitindo fácil manutenção e adaptação às necessidades emergentes. Tal arquitetura constitui um *sistema híbrido* [30]. Neste caso, a tarefa maior cabe à interface entre os módulos lógico/numéricos e ao processo de aquisição de conhecimento [31].

Na busca por modelos mais eficientes, o programa SINTIA (*Síntese Interativa da Transmissão / Inteligência Artificial* [31]) - na verdade, um protótipo - se mostrou mais interativo e adaptativo do que seus antecessores. Por incorporar técnicas de IA, possui as heurísticas de planejamento separadas do resto do modelo, o que facilita sua manutenção, uma vez que a parte lógica opera em separado das rotinas de simulação.

Para melhor apreciação do apoio fornecido por estes modelos ao especialista em planejamento, os próximos itens descrevem os processos de síntese estática e dinâmica, sob uma forma conceitual.

2.4. O Processo de Síntese de Sistemas de Transmissão

O objetivo do processo de síntese de sistemas de transmissão é estabelecer um cronograma de obras de expansão dentro de um horizonte de estudos pré-fixado. Para tanto, o sistema é visto em regime permanente, sob condições normais (vide Quadro 2.1) e o processo de síntese se desenvolve em duas etapas, quais sejam, a síntese estática e a síntese dinâmica, descritas a seguir.

2.4.1. Síntese estática

Como prática nas empresas do setor de energia elétrica, o objetivo do processo de síntese estática é sanar os eventuais problemas de sobrecarga ou violação de tensão na rede em estudo,

considerando fixos os valores de demanda e geração para cada estágio de planejamento, dentro do horizonte de estudos a analisar.

A Figura 2.6 descreve o algoritmo conceitual para síntese estática de redes de transmissão.

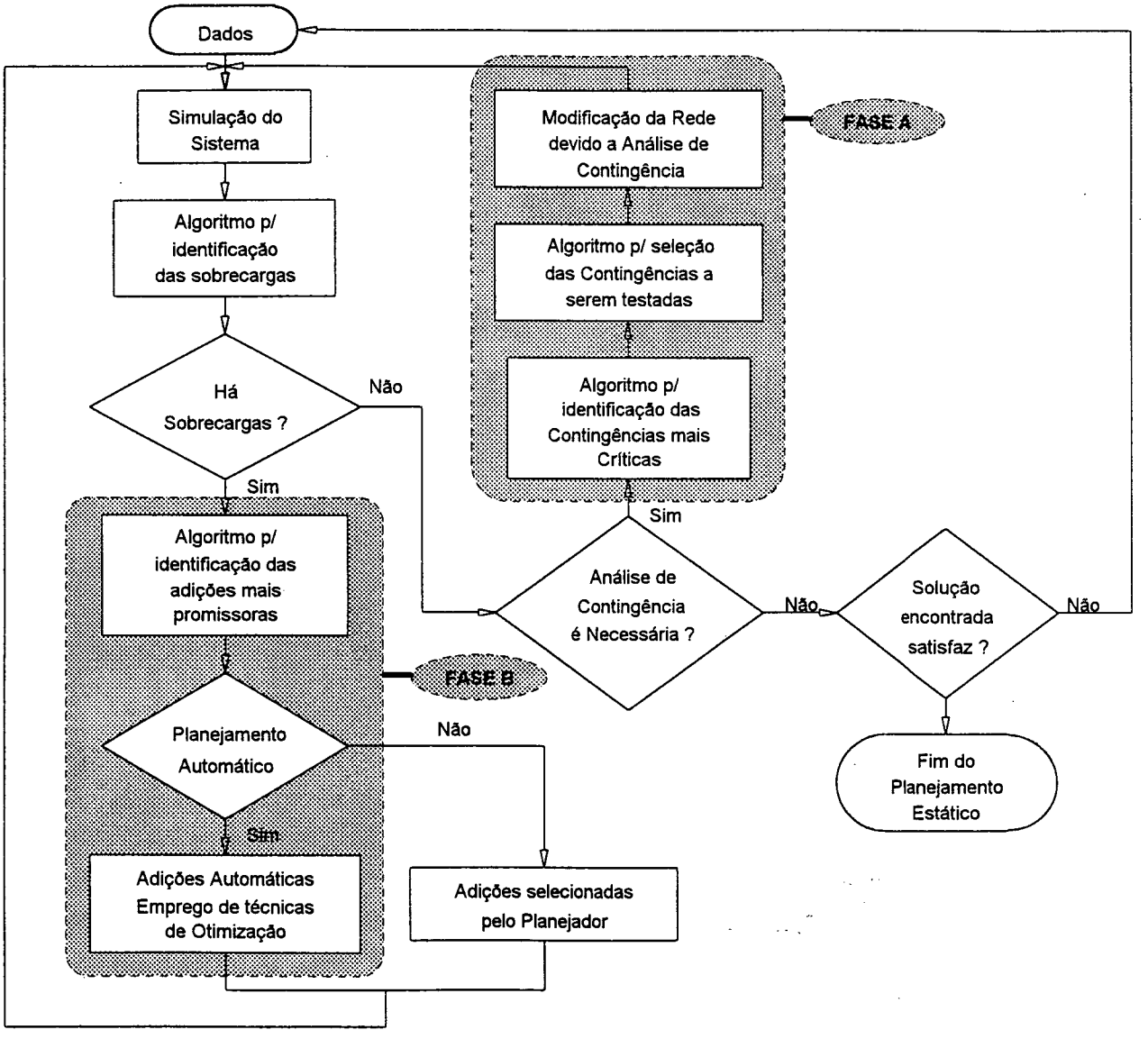


FIGURA 2.6 - ALGORITMO CONCEITUAL PARA SÍNTESE ESTÁTICA DE REDES DE TRANSMISSÃO

O estudo inicia pela simulação do sistema e análise em regime permanente com fluxo de potência sob condições normais (vide Quadro 2.1). Tal análise visa detectar situações anormais

(sobrecarga nos ramos de transmissão*, violações de tensão nos barramentos, etc.) que comprometam, a longo prazo, o transporte confiável de energia elétrica no sistema de potência em estudo. Se houver, passa-se a uma de duas formas de solucionar o problema, levando-se em conta, ou não, a análise de contingências - isto está representado na Figura 2.6 pelas hachuras *Fase A* e *Fase B*, respectivamente. O processo continua até que se encontre uma solução satisfatória, onde o critério de parada fica a cargo do usuário. Assim, pode ser que ainda haja situações anormais, mas estas podem estar dentro de um nível aceitável (sobrecarga de 2%, por exemplo), ou ainda pode ser que haja uma solução melhor a considerar (solução ótima). Este algoritmo ressalta a prática do especialista em planejamento, o qual busca soluções plausíveis e não soluções ótimas, difíceis de se conseguir em sua totalidade.

2.4.2. Síntese dinâmica

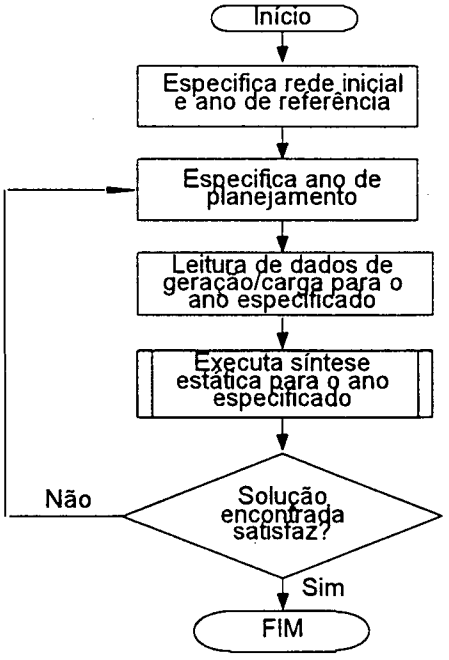


FIGURA 2.7 - ALGORITMO CONCEITUAL PARA SÍNTESE DINÂMICA DE REDES DE TRANSMISSÃO

* Linhas de transmissão, transformadores, elos de CC, etc. [7]

Tomando por base o algoritmo da Figura 2.6, a Figura 2.7 representa o algoritmo conceitual para síntese dinâmica de redes de transmissão.

O plano global de expansão é abordado de forma dinâmica pelo planejador, através do acompanhamento dos efeitos que a síntese estática de um estágio exerce sobre os estágios subsequentes. Busca assim, definir os investimentos de base e o grau de influência que a escolha atual exerce sobre os estágios seguintes.

Como já foi mencionado anteriormente, o processo combinatório envolvido na criação de planos de expansão impede a análise de todas as alternativas de expansão possíveis. Algumas configurações seriam por si só absurdas - tal como adicionar todos os circuitos previstos para a rede em estudo, inclusive suas duplicações. Assim, apenas um número limitado de configurações pode ser analisado.

2.5. Estratégias de Expansão Alternativas

Em vista da complexidade no trato com heurísticas de planejamento da expansão, pode-se considerar duas estratégias básicas de estudo, quais sejam, FORWARD e BACKWARD [57]. Na estratégia FORWARD, o problema da expansão é abordado de forma sequencial. Esta abordagem é repetida desde o ano inicial até o ano horizonte. O método de ano horizonte é uma variante da estratégia FORWARD, na qual são considerados para expansão apenas os circuitos que comparecem na configuração do ano horizonte. A estratégia FORWARD corresponde a uma estratégia de expansão incremental (Figura 2.8).

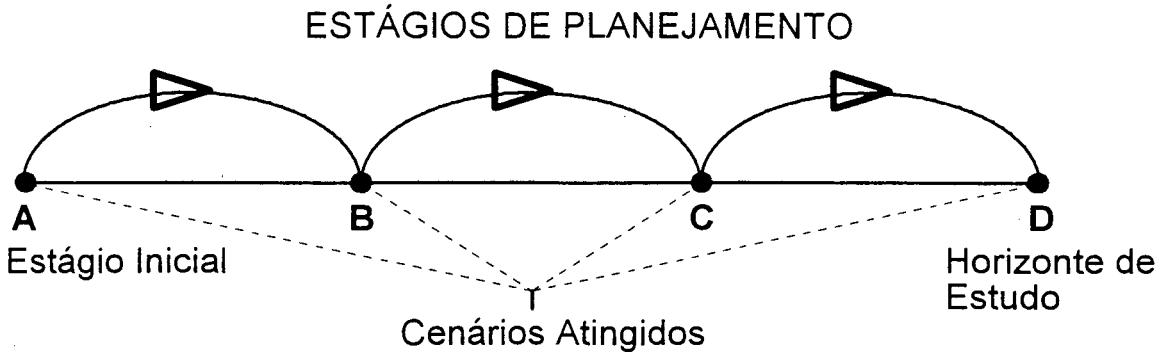


FIGURA 2.8 - ESTRATÉGIA FORWARD

Na estratégia BACKWARD, a primeira etapa consiste em realizar a síntese estática para o ano horizonte, enquanto que o planejamento para os estágios intermediários é realizado no sentido inverso do tempo, isto é, a partir do horizonte de estudo até o estágio inicial, como mostra a Figura 2.9. A cada estágio, a lista de candidatos à adição é limitada aos circuitos que já foram adicionados no *ano alvo* (ano horizonte). Desta forma, reforços efetuados em estágios intermediários são antecipações de circuitos cuja adição seria necessária em anos posteriores [7]. Na verdade, a estratégia BACKWARD consiste na aplicação do conceito de ano horizonte juntamente com o de expansão dinâmica.

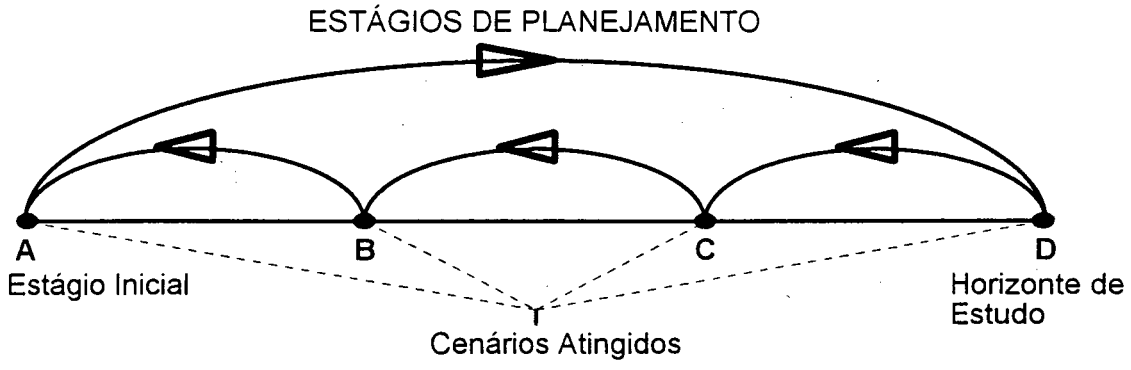


FIGURA 2.9 - ESTRATÉGIA BACKWARD

O objetivo dos planos de expansão é sanar eventuais sobrecargas à medida que se manifestem no decorrer do processo. Na estratégia FORWARD, o futuro é sempre visto como uma *caixa preta*. Na estratégia BACKWARD, o único *tiro no escuro* ocorre para o ano horizonte.

A escolha de uma ou outra estratégia pode implicar em economias consideráveis no planejamento de redes. Estudos com sistemas testes constataram a otimização de expansões utilizando-se a estratégia BACKWARD. Esta abordagem *dinâmica*, ao limitar o número de candidatos à adição, permite tratar redes de maior porte e, por usar como alvo a solução do ano horizonte, viabiliza a antecipação de circuitos e grandes economias de escala, o que permite a obtenção de planos de expansão mais realistas [7].

2.6. CONCLUSÕES

Este capítulo procurou dar uma visão geral sobre o processo de planejamento de sistemas de transmissão, abordando o processo de planejamento e o estado da arte em modelos de síntese da expansão.

Com esta visão de sistemas de potência, é possível intuir o esforço e a responsabilidade dos engenheiros envolvidos no processo de decisão sobre alternativas de expansão. Como o próprio texto procurou ressaltar, o uso de modelos computacionais visa apenas dar um melhor apoio ao planejador, uma vez que a decisão final ficará sempre a cargo do elemento humano [26].

A análise da literatura mostra que as diversas abordagens propostas apresentam soluções parciais do problema de planejamento da transmissão. Mostra ainda que o conhecimento e a experiência do planejador constitui elemento essencial de um processo de solução satisfatório, ainda que não garanta o *ótimo* (o qual ainda não foi definido de forma precisa).

As tentativas já realizadas no sentido de incorporar o conhecimento do planejador ainda são limitadas em conteúdo e forma de implementação, dependendo da modificação de códigos extremamente especializados para se incluir novas heurísticas. A utilização de técnicas de inteligência artificial se apresenta como uma via de aperfeiçoamento promissora, como será discutido nos próximos capítulos.

CAPÍTULO 3 - EVOLUÇÃO DOS PARADIGMAS DE PROGRAMAÇÃO

"O homem deve herdar do homem, de vez que os seus trabalhos e resultados são coletivos."

O Evangelho Segundo o Espiritismo
Cap XX - Henrique Heine, Paris 1863

3.1. INTRODUÇÃO

Até agora, esta dissertação se ocupou em conceituar o processo de planejamento, mostrando que parte das limitações encontradas nos modelos atuais decorre da forma convencional de programar. Devido a estas limitações, várias pesquisas têm sido dirigidas em busca da definição de novos paradigmas de programação. O presente capítulo dará uma introdução a este tema, situando cronologicamente a evolução dos paradigmas de programação utilizados no setor elétrico.

Os primeiros conceitos sobre *Inteligência Artificial* (IA) surgem em meados da década de 50, trazendo consigo a proposta de estudar meios capazes de tornar os computadores tão eficientes quanto um ser humano na execução de uma dada tarefa.

Até o final da década de 60, as pesquisas em IA foram conduzidas objetivando o desenvolvimento de mecanismos formais de inferência que fossem genéricos e pudessem ser aplicados a diferentes problemas de diversas áreas de pesquisa [72].

O mecanismo de solução se resumia a explorar o conjunto de combinações dos parâmetros envolvidos no problema, gerando e comparando todas as soluções possíveis, até encontrar a melhor. Assim, esta metodologia logo se mostrou ineficiente, tendo em vista que o custo

computacional para se fazer frente à explosão combinatorial gerada neste processo é muito elevado [68].

A partir dos anos 70, à medida que as pesquisas em IA progrediam, observou-se ser fundamental estudar sistemas mais específicos em detrimento de sistemas genéricos. Os estudos conduziram ao detalhamento de diversos fenômenos da natureza, em busca dos seus elementos constituintes. Isto alargou a área de aplicação da computação, formalizando tais fenômenos em blocos altamente especializados e independentes, modularizando e especializando o conhecimento.

Este método levou à implementação prática de diversos sistemas, cujas áreas de atuação abrangem a compreensão de linguagem natural, a robótica e o reconhecimento de imagens, bem como a análise, diagnóstico e resolução de problemas específicos nos campos de medicina, álgebra, engenharia, química, entre outros, como mostra o Quadro 3.1 [31].

QUADRO 3.1 - APLICAÇÕES GERAIS DE SISTEMAS DE CONHECIMENTO

NOME	ÁREA	FUNÇÃO
CASNET	medicina	diagnose e tratamento de glaucoma
DENDRAL	química	elucidação de estrutura molecular
DIPMETER ADVISER	geologia	exploração de petróleo
EL	engenharia elétrica	análise de circuitos elétricos
EMYCIN	engenharia de conhecimento	construção de sistemas de conhecimento
INTERNIST	medicina	diagnóstico de medicina interna
MYCIN	medicina	diagnóstico e tratamento de infecções no sangue
PROSPECTOR	geologia	exploração mineral
R1	computação	configuração de computadores
TEIRESIAS	medicina	aquisição de conhecimento
MACSYMA ADVISOR	matemática	manipulação de expressões algébricas
PUFF	medicina	tratamento de doenças pulmonares
SINTIA	engenharia	planejamento da expansão de sist. de potência

Na área de sistemas de potência, o desenvolvimento de métodos de planejamento com apoio de computadores progrediu como mostra a Figura 3.1 [9].

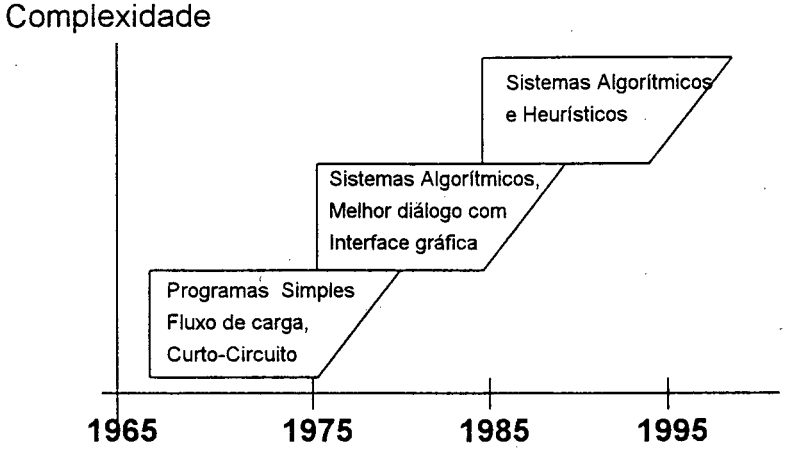


FIGURA 3.1 - DESENVOLVIMENTO DE MÉTODOS DE PLANEJAMENTO COM APOIO DO COMPUTADOR

Na primeira fase de desenvolvimento, só era possível investigar sistemas de potência de tamanho limitado. O desenvolvimento de técnicas matemáticas para tratar com matrizes esparsas [49], juntamente com o incremento na performance dos computadores, tornou possível a estes programas lidar com redes bem maiores.

Comparados com a situação atual, os computadores primitivos apresentavam fraco desempenho por unidade e alto custo de investimento. Na maior parte dos casos, eram *mainframes* e executavam operações com grupos de multiusuários. Longos períodos de espera e um grande esforço para a preparação e subsequente avaliação das alternativas eram características comuns na primeira fase de métodos apoiados por computadores.

Na segunda fase de desenvolvimento, computadores mais avançados, com vídeos de alta resolução estavam disponíveis. Isto reduziu o tempo de resposta e melhorou a comunicação homem-máquina (interface com o usuário). Os programas foram integrados em sistemas de programas, e os resultados podiam ser mostrados diretamente no monitor. Os métodos de engenharia foram evoluindo de simples algoritmos para métodos de simulação e otimização, mais complexos. Isto permitiu a automação das rotinas de trabalho dos especialistas. Exemplos deste

desenvolvimento são os programas de fluxo de carga com análise automática de contingências e os programas para otimização da expansão de sistemas de geração considerando custos, confiabilidade e limites técnicos.

A Figura 3.2 mostra a estrutura de um sistema da segunda fase. O usuário pode comunicar com o computador via teclado e monitor colorido. Ele define as variáveis, inicia a simulação e é suprido com representações gráficas dos resultados. O desenvolvimento de variantes e interpretação dos resultados é feito principalmente pelo usuário.

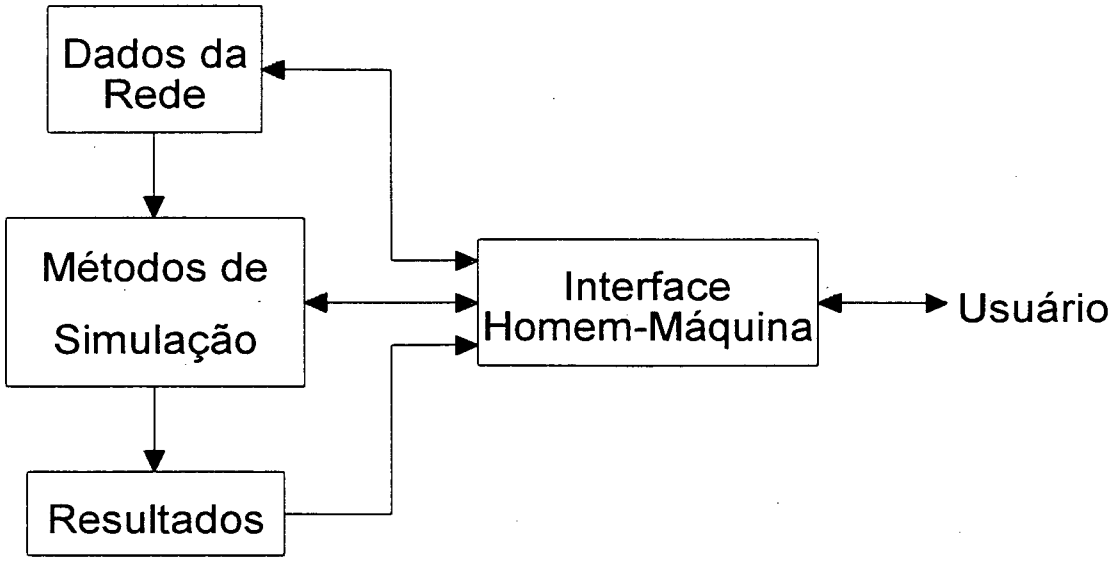


FIGURA 3.2 - ESTRUTURA DE UM SISTEMA DA SEGUNDA FASE

Os métodos de engenharia usados na segunda fase podem ser assim categorizados:

a) Os métodos usados para resolver os problemas são algorítmicos;

b) As soluções determinísticas são dominantes. Soluções heurísticas, baseadas na experiência de especialistas, não são comuns;

c) Os programas são relativamente inflexíveis com respeito a alterações das condições; como os algoritmos, tem de ser adaptados;

d) O trabalho necessário para desenvolvimento é grande e pode exigir vários homens-ano para uma complexa ferramenta de *software*;

e) Embora sistemas de potência de grande porte possam ser investigados, somente problemas de limitada complexidade podem ser solucionados.

Atualmente, o desenvolvimento de métodos de engenharia com apoio de computadores está no começo da terceira fase, a qual está influenciada pelos métodos baseados no conhecimento. As seguintes mudanças têm provado ser necessárias:

a) Os métodos baseados no conhecimento complementam os métodos puramente algoritmicos. Os algoritmos ainda são necessários para simulação de sistemas de potência. Sua principal aplicação está na solução de um problema complexo, que não mais pertence ao algoritmo, mas está descrito na forma de regras de produção;

b) Como estas regras agora não fazem parte do programa, a adaptação ou alteração nas precondições é mais flexível. Em muitos casos, basta alterar ou suplementar regras e fatos na base de conhecimento;

c) A solução de um problema não é feita exclusivamente de acordo com os critérios a serem matematicamente resolvidos, mas com o auxílio de algoritmos estruturados e heurísticas de planejamento, o que permite o uso de conhecimentos que especialistas adquiriram com o manejo de eventos reais.

d) Sistemas baseados no conhecimento podem ser usados para análise e síntese de acordo com as regras fornecidas. Assim, elas são adaptadas para encontrar fragilidades, padrões de comportamento, tendências, bem como para propor planos de expansão.

Nos programas da terceira fase, as tendências levam a crer na possibilidade dos métodos de análise serem integrados a ferramentas de sistemas baseados no conhecimento, tendo o mecanismo de inferência como parte vital do sistema (Figura 3.3).

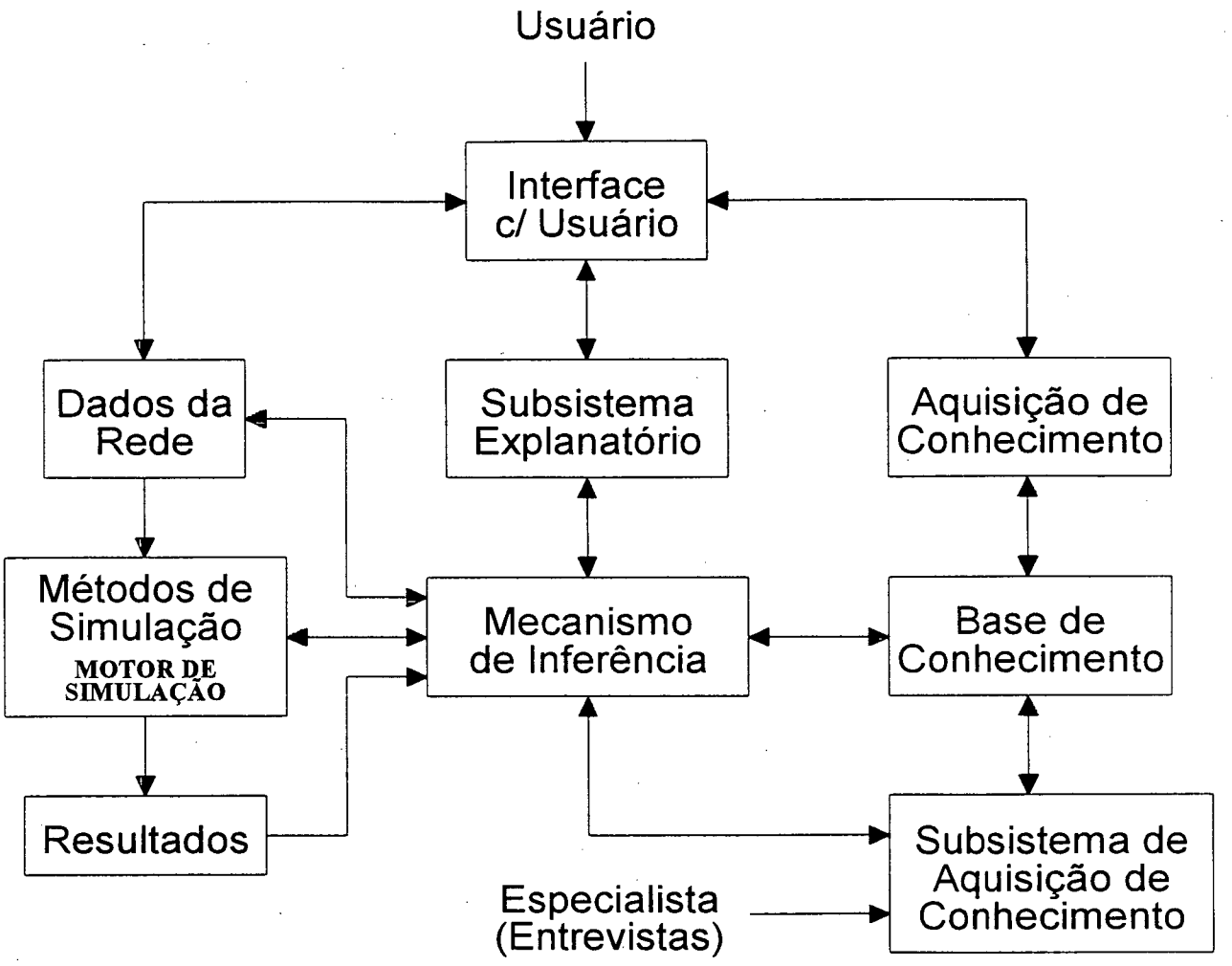


FIGURA 3.3 - ARQUITETURA DE UM PROGRAMA BASEADO NO CONHECIMENTO

A Figura 3.3 mostra a arquitetura típica de um programa baseado no conhecimento e pode ser assim interpretada:

a) O Mecanismo de Inferência controla o fluxo de informações de três formas, de acordo com as regras e os fatos contidos na base de conhecimento:

   i. pela seleção do método analítico mais apropriado;

   ii. pela definição de novos valores para os dados da rede;

   iii. pela filtragem dos resultados apresentados pelo motor de simulação, eliminando soluções espúrias;

b) Novos dados de entrada podem ser estimados como resultados de processos de inferência efetivados em conjuntos de regras implementados. Por causa desta inferência, certas variáveis podem ser alteradas e os dados dos elementos, o *tap* de transformação ou a carga da rede, e os resultados podem ser interpretados automaticamente;

c) Caso necessário, um subsistema explanatório informa o usuário sobre o estado das variáveis de simulação, seus valores e o processo de inferência que levou aos mesmos. Isto é de grande utilidade na fase inicial de um sistema baseado no conhecimento, sendo uma forma de permitir que o especialista verifique se a base de regras está completa, além de facilitar a familiarização do usuário com a mesma.

No sistema mostrado, tanto o especialista quanto o usuário têm acesso à base de conhecimento. Em alguns sistemas, somente o especialista tem habilidade suficiente para alterar esta base.

Há uma grande tendência na busca por soluções híbridas que facilitem a criação de ferramentas para a área de engenharia [30]. Tais sistemas incorporariam a confiabilidade e robustez de algoritmos numéricos disponíveis no mercado, com a flexibilidade oferecida pelas técnicas de sistemas baseados no conhecimento e programação orientada a objeto.

Para muitas aplicações, um sistema híbrido tem um desempenho melhor em comparação ao de um algoritmo puramente numérico.

Na evolução dos paradigmas de programação, o que se pode notar não é apenas a estrutura dos modelos finais, mas a própria filosofia empregada no seu desenvolvimento. No que diz respeito a sistemas baseados no conhecimento e sistemas orientados a objeto, a questão central diz respeito à definição da base de conhecimentos, ou seja, a estipulação dos elementos chave que melhor descrevam o problema a ser modelado.

Nos últimos cinco anos, foram desenvolvidos diversos aplicativos empregando ferramentas de IA em sistemas de potência [60]. Tais implementações foram levadas a efeito tanto por indústrias ligadas ao setor elétrico, quanto por concessionárias de energia, laboratórios de pesquisa e universidades (Figura 3.4).

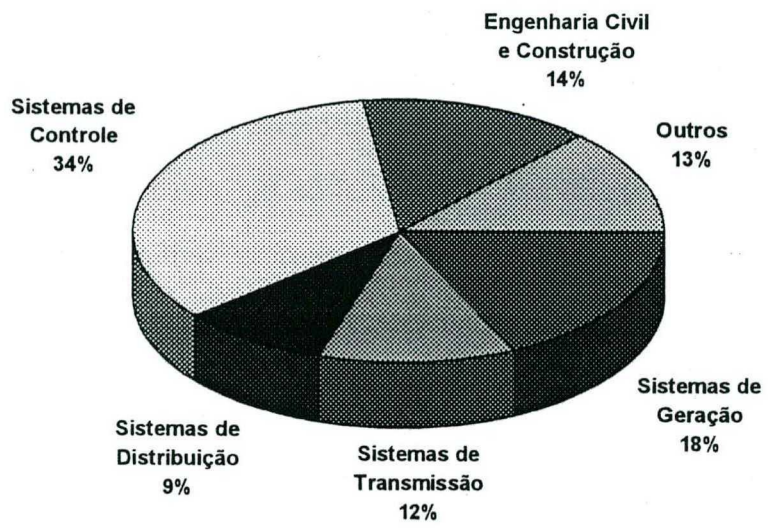


FIGURA 3.4 - LINHAS DE PESQUISA DE IA EM SISTEMAS DE POTÊNCIA

3.2. PROGRAMA CONVENCIONAL *x* SISTEMAS BASEADOS NO CONHECIMENTO

O Quadro 3.2 faz um comparativo entre programas convencionais e sistemas baseados no conhecimento [72].

QUADRO 3.2 - PROGRAMA CONVENCIONAL X SISTEMA BASEADO NO CONHECIMENTO

PROGRAMA CONVENCIONAL	SISTEMA BASEADO NO CONHECIMENTO
Representação e uso de dados	Representação e uso de conhecimento
Algorítmico	Heurístico
Processo repetitivo	Processo de inferência
Manipulação de bases de dados	Manipulação de bases de conhecimento

Enquanto um método matemático garante a produção de um solução correta ou ótima, um método heurístico produz uma solução aceitável para a maioria das situações.

Sistemas baseados no conhecimento diferem de programas convencionais nos seguintes aspectos:

a) Os programas convencionais são algorítmicos, enquanto os sistemas baseados no conhecimento são declarativos, fazendo uso de heurísticas na modelagem do mesmo;

b) Existe uma separação física entre os diversos componentes funcionais de um sistema baseado no conhecimento. A base de conhecimentos, o mecanismo de inferência e o motor de simulação são contidos em módulos independentes entre si, característica que não se verifica em sistemas convencionais;

c) Como consequência dos itens (a) e (b) acima, novas heurísticas podem ser adicionadas numa base de conhecimentos sem que o mecanismo de inferência seja afetado. Desta forma, o risco de se cometer enganos na modelagem que possam provocar uma eventual falha do programa, diminuem;

d) A divisão da base de conhecimentos em módulos à parte, constituindo a base de fatos e a base de dados, facilita sua compreensão e eventuais modificações;

e) Os programas convencionais são orientados para o processamento numérico, enquanto que os sistemas baseados no conhecimento têm sido utilizados principalmente para o processamento simbólico;

f) Conforme exposto anteriormente e exemplificado através da Figura 3.3, os sistemas baseados no conhecimento permitem a geração de mensagens relativas às inferências utilizadas na solução do problema. Esta capacidade é mais difícil de ser implementada nos sistemas convencionais.

As principais vantagens que as técnicas de sistemas baseados no conhecimento oferecem sobre a programação convencional estão listadas a seguir*:

a) Aumento de produtividade no desenvolvimento de protótipos e nos estudos preliminares de viabilização dos sistemas;

b) Diminuição dos custos de desenvolvimento;

c) Simplificação da tarefa de manutenção de sistemas, uma vez que a modificação da base de conhecimentos pode ser feita com relativa rapidez quando for necessário adaptar mudanças no modelo do problema a ser simulado.

Em um programa convencional, o controle do fluxo e uso dos dados é pré-determinado pelo código do programa. O processamento é feito de forma sequencial e a mudança de direção ocorre em pontos específicos. Programas convencionais são úteis para resolver problemas tais como a solução de um conjunto de equações lineares.

Para problemas decorrentes da manipulação dos dados, SBC oferecem a oportunidade de examinar o estado do sistema em estudo a cada passo e propor uma ação apropriada. O uso de SBC também simplifica o trabalho de explicar o que o programa fez ou como ele chegou a uma particular conclusão.

---

* Tais vantagens também valem para programas orientados a objeto, uma vez que seus paradigmas descendem das técnicas de sistemas baseados no conhecimento representados na forma de *frames* (ver [72]; ver também Capítulos 4 e 5).

3.3. PARALELO ENTRE ESPECIALISTAS HUMANOS E SISTEMAS DE CONHECIMENTO

O Quadro 3.3, abaixo, ilustra algumas diferenças e singularidades entre o uso de *Especialistas Humanos* e *Sistemas Baseados no Conhecimento* do ponto de vista do conhecimento de cada um [72].

QUADRO 3.3 - VANTAGENS DE SISTEMAS BASEADOS NO CONHECIMENTO SOBRE ESPECIALISTAS HUMANOS

SISTEMA BASEADO NO CONHECIMENTO	ESPECIALISTA HUMANO
Permanente	Perecível
Fácil de transmitir	Difícil de transmitir
Fácil de documentar	Difícil de documentar
Consistente	Imprevisível
Econômico	Dispendioso financeiramente

Contudo, apesar das vantagens inerentes a sistemas de conhecimento, ainda é necessário manter o elemento humano no controle de qualquer processo. O Quadro 3.4 aponta as vantagens e singularidades de especialistas humanos em detrimento do conhecimento artificial.

QUADRO 3.4 - VANTAGENS DE ESPECIALISTAS HUMANOS SOBRE SISTEMAS BASEADOS CONHECIMENTO

ESPECIALISTA HUMANO	SISTEMA BASEADO NO CONHECIMENTO
Criativo	Sem inspiração
Adaptativo	Necessita explicitar o que fazer
Experiência sensorial	Dados de entrada simbólicos
Amplo foco de atenção	Estreito foco de atenção
Conhecimento de consenso	Conhecimento técnico

3.4. CONCLUSÕES

O presente capítulo procurou apresentar a evolução das técnicas de inteligência artificial desde os programas convencionais até os presentes sistemas baseados no conhecimento. Paralelos foram feitos entre os mesmos ressaltando os limites de aplicabilidade que os mesmos possuem.

O modelo TANIA foi uma primeira tentativa de utilizar a IA no planejamento de sistemas de transmissão. Ficou limitado a sistemas de pequeno porte [28], além de se mostrar inflexível para inclusão de novas heurísticas.

Como modelo interativo, o SINTRA [7] se tornou capaz de atingir resultados mais satisfatórios do que o TANIA. Apesar de também não ser flexível para inclusão de novas heurísticas, permitiu que o planejador empregasse estratégias de planejamento (vide Capítulo 2), como *forward* e *backward*, que o encaminhavam a um resultado aceitável.

O modelo SINTIA [31], foi um dos primeiros protótipos de sistema baseado no conhecimento para síntese da expansão de sistemas de transmissão. Para isso, utilizou diversas linguagens de programação (PASCAL, FORTRAN, PROLOG e ASSEMBLER) integradas num modelo único, formando um sistema baseado no conhecimento flexível e modularizado.

Os próximos capítulos apresentarão, com mais detalhes, a forma conceitual de sistemas baseados no conhecimento e programação orientada a objeto, constituindo-se em importantes subsídios para o Capítulo 6, responsável pela descrição da metodologia de desenvolvimento usada para implementar o protótipo SAPIENS, um sistema experimental de apoio ao planejamento que une conceitos de inteligência artificial com rotinas convencionais de simulação.

CAPÍTULO 4 - SISTEMAS BASEADOS NO CONHECIMENTO

4.1. INTRODUÇÃO

Os capítulos precedentes descreveram a evolução das ferramentas computacionais comumente utilizadas para estudos de planejamento de sistemas de potência. Tal processo vem ocorrendo em várias áreas de pesquisa, conforme ressaltado no último capítulo, sendo marcantes os efeitos que as técnicas de inteligência artificial têm causado no meio científico.

Assim, o presente capítulo irá tratar do paradigma de programação baseado no conhecimento.

A Figura 4.1 ilustra a organização de sistemas de inteligência artificial, situando os sistemas baseados no conhecimento como uma categoria especializada dos mesmos.

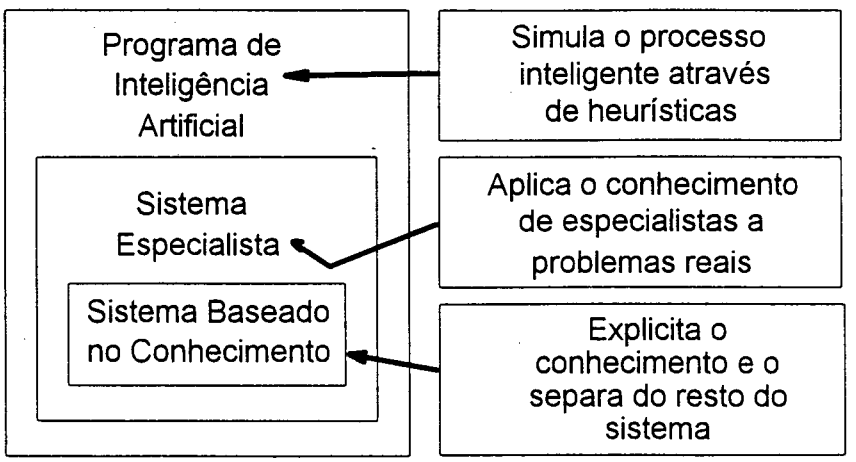


FIGURA 4.1 - SISTEMAS DE IA

Como mostra a Figura 4.1, existe um pequena diferença entre sistemas especialistas e sistemas baseados no conhecimento. Antes de descrever tal diferença, é preciso definir o que vem a ser um sistema especialista.

A resposta a esta questão pode ser obtida ao observar o comportamento de um especialista humano, uma pessoa que adquiriu, ao longo de sua carreira profissional, habilidade suficiente para resolver certos problemas. Primeiramente, ele analisa o problema, tentando definir os mecanismos que regem sua solução. Tomando por base o conhecimento à disposição, começa a estipular premissas, regras de trabalho, baseadas em fatos atualizados e nas respostas obtidas. Assim, com a prática adquirida no exercício da profissão, desenvolve heurísticas, que lhe indicam o caminho a seguir para resolver certo problema.

Assim, por analogia, sistemas especialistas são programas computacionais capazes de atuar tão bem quanto um especialista de uma certa área. Em sua modelagem, tanto as premissas quanto os fatos relevantes à análise de uma dada situação são ditados pelo especialista em questão.

O termo *sistema especialista* teve origem na forma com que os primeiros programas deste tipo foram construídos, uma vez que a captura e síntese dos conhecimentos eram realizadas através de entrevistas com especialistas de renome na área de interesse (vide Figura 4.2).

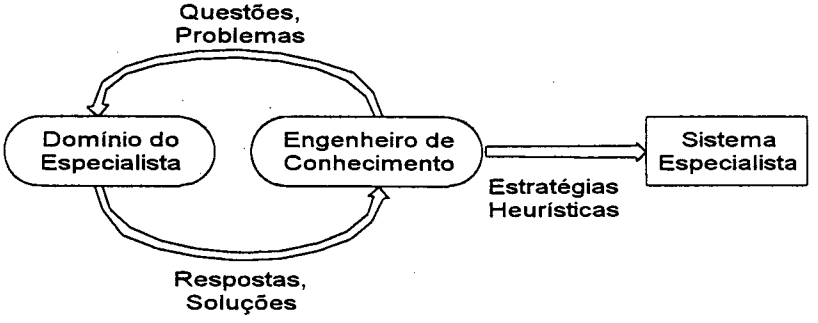


FIGURA 4.2 - AQUISIÇÃO DE CONHECIMENTO

O termo adotado atualmente para este tipo de programa é *sistema baseado no conhecimento*, devido ao fato de que nem todo sistema especialista é desenvolvido para alcançar um desempenho equivalente ao de um especialista humano. Na maior parte das vezes, estes

programas são implementados com conhecimento suficiente apenas para auxiliar o usuário durante o processo decisório para o qual foram projetados.

Não se espera que tais sistemas sejam completos, ou autônomos. Espera-se, isso sim, que os mesmos sirvam para modelar o conhecimento básico de interesse para uma dada área, ou seja, para verificar a redundância de fatos e premissas nas heurísticas dos especialistas entrevistados. Assim, regras espúrias seriam eliminadas dos processos usuais.

O emprego que se faz deste conhecimento não é unipolarizado, destinando-se a vários fins, como por exemplo, treinamento de pessoal, validação de processos (eliminação de regras espúrias), modelagem preditiva (diagnóstico), armazenamento de conhecimento (memória institucional), etc.

A Figura 4.3 ilustra a organização de um sistema baseado no conhecimento [31], mostrando seus componentes básicos, quais sejam, a base de conhecimentos e o mecanismo de inferência. A separação entre o conhecimento e o resto do corpo do sistema visa facilitar o desenvolvimento de procedimentos para manipular este conhecimento [72].

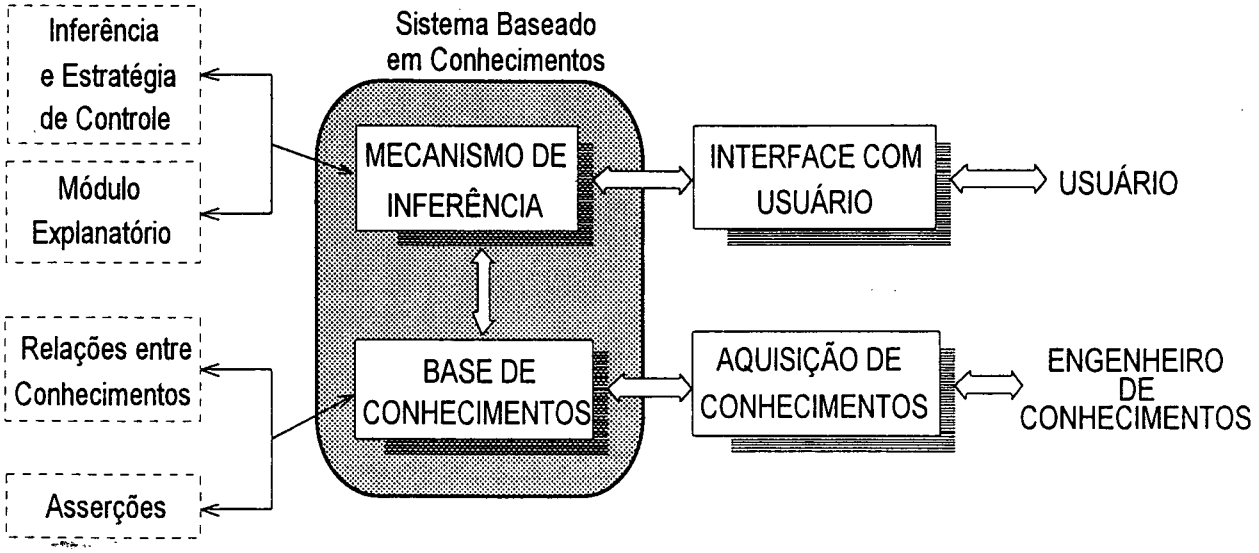


FIGURA 4.3 - ARQUITETURA DE UM SISTEMA BASEADO NO CONHECIMENTO

4.2. REPRESENTAÇÃO DO CONHECIMENTO

Os estudos de IA têm por objetivo modelar as facilidades mentais humanas, permitindo às máquinas executarem suas tarefas de uma forma mais inteligente. Implícito neste objetivo está também a preocupação de codificar, preservar e explorar os conhecimentos adquiridos pelo homem, uma vez que os especialistas humanos são perecíveis.

Como citado em [72],

> *"para fazer um programa inteligente, providencie o maior volume possível de conhecimento sobre uma dada área de estudo."*

Tais esquemas buscam modelar a forma como o ser humano raciocina ao resolver diversos problemas. Em outras palavras, suas heurísticas. Segundo Feigenbaum [29]:

> *"Os conhecimentos de um sistema baseado no conhecimento consistem de fatos e heurísticas. Os fatos constituem um conjunto de informações que é extensivamente compartilhado, publicamente disponível e endossado pelos peritos da área de uma forma geral. As heurísticas são regras de julgamento com base em bom senso (regras de raciocínio plausível, regras de boa estimativa, etc.), de uso privado e menos debatidas, as quais caracterizam a tomada de decisão a nível do especialista da área. O desempenho de um sistema de conhecimento é em função principalmente da extensão e qualidade de seus conhecimentos."*

Referindo-se à Figura 4.3, a base de conhecimentos pode ser vista como um repositório de informações relativas ao problema a ser resolvido, as quais podem ser classificadas em duas categorias: *asserções* e *relações entre conhecimentos*.

As asserções correspondem aos *fatos* da definição de Feigenbaum e descrevem não somente o problema proposto, mas também o atual estágio de solução em que o mesmo se encontra. Tais informações também são denominadas de dados temporários ou dados de memória de trabalho,

uma vez que podem ser criadas ou destruídas dinamicamente no decorrer do processo de inferência.

As relações entre conhecimentos são fórmulas que expressam as ligações existentes entre as diversas asserções do problema. O tipo de fórmula mais comum é a regra de produção, que tem o seguinte formato:

*Se **premissas** Então **conclusão.***

Durante a solução de um problema, as asserções e as regras são utilizadas pelo mecanismo de inferência para gerar conclusões lógicas. O processo de inferência é feito através de formalismos matemáticos denominados modelos de inferência e estratégias de controle. Um modelo de inferência tem a função de examinar o conteúdo da base de conhecimentos, identificando, a cada passo, regras cujas premissas podem ser satisfeitas pelas asserções existentes. A aplicação de uma regra ao problema corresponde à adição da respectiva parte conclusiva à base de conhecimento, o que cria uma nova asserção que possibilita o prosseguimento do processo de inferência.

O objetivo das estratégias de controle é estabelecer uma sequência ordenada de inferências, de modo a guiar o processo de solução em direção à uma resposta final.

Uma característica adicional do mecanismo de inferência é a habilidade de *justificar* a *linha de raciocínio* utilizada ao resolver o problema. Isto pode ser feito através da reconstituição da sequência de regras *disparadas* durante o processo de inferência. Contudo, é bom que se note que o módulo explanatório não é capaz de fornecer explicações sobre o processo de solução do problema em si. Simplesmente é feita uma composição das mensagens associadas às regras aplicadas com sucesso no mesmo.

O mecanismo de inferência pode solicitar do usuário, em tempo de execução, informações adicionais sobre o problema quando a base de conhecimentos se mostrar incompleta e/ou o processo de inferência assim exigir.

A codificação e acumulação do conhecimento é um dos aspectos mais importantes de um sistema baseado no conhecimento. Os técnicos encarregados de extrair, sistematizar e codificar o conhecimentos são chamados *engenheiros de conhecimento*, e a metodologia de trabalho que estes empregam é denominada *aquisição de conhecimento*, como mostra a Figura 4.2 [72]. Como se pode inferir da mesma, o processo de aquisição de conhecimento é realimentado, a fim de que os especialistas envolvidos acompanhem a evolução dos protótipos implementados, e possam apreciá-los do ponto de vista de sua viabilidade dentro do escopo estipulado.

Pela experiência adquirida com o protótipo SAPIENS, o engenheiro de conhecimento tem que pensar como um especialista, para compreender as heurísticas com que trava contato. Tanto o especialista entrevistado quanto o engenheiro de conhecimento devem ter em mente que um sistema baseado no conhecimento é implementado com o fim de auxiliar o usuário no processo decisório. Assim, e pelas limitações apresentadas no Capítulo 3, Quadros 3.3 e 3.4, a decisão final neste tipo de sistema deve ficar sempre a cargo do usuário.

Existem muitas técnicas para representação de conhecimento. As três formas mais empregadas são [72]:

a) regras de produção;

b) redes semânticas;

c) frames.

4.2.1. Regras de produção

De modo geral, a etapa inicial na solução de um problema envolve sua completa descrição. Um meio eficiente de efetuar esta tarefa é através da representação do espaço de estados que o referido problema pode gerar, ou seja, as configurações relevantes e as premissas que controlam

as transições entre estados. Este tipo de representação estabelece uma relação de correspondência entre o processo de solução de problemas e busca em espaço de estados [28].

Em muitos programas *inteligentes*, o processo de busca constitui um meio indispensável à solução do problema. Para facilitar este processo, os sistemas de inteligência artificial são estruturados segundo um formalismo computacional denominado sistema de produção, cuja constituição é mostrada na Figura 4.4.

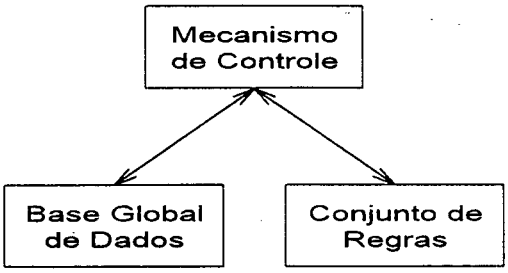


FIGURA 4.4 - ESTRUTURA DE UM SISTEMA DE PRODUÇÃO

Este método lida basicamente com *premissas* e *conclusões*. Estas últimas surgem como novo fato que é posteriormente inferido na base de conhecimentos. Suas declarações são do tipo:

If *premisse* **Then** *conclusion*.

A base de dados é utilizada para armazenar as informações relativas ao estado atual do problema, enquanto que as regras de produção são empregadas como agentes transformadores da base de dados no decorrer da solução. O mecanismo de controle monitora a sequência de aplicação de regras segundo uma estratégia pré-estabelecida, interrompendo a computação quando o conteúdo da base atender às condições de término do problema.

Os sistemas de produção funcionam segundo o algoritmo abaixo [31]:

1. Considere uma base de dados contendo as informações do estado inicial do problema.

2. Se a base de dados satisfaz a condição de término do problema, então FIM.

3. Caso contrário, selecione uma regra (R) através da estratégia de controle.

4. Aplique a regra R ao estado atual do problema.

5. Atualize a base de dados.

6. Vá para 2.

O uso de sistema de produção oferece algumas vantagens, entre elas:

(a) A estrutura de um sistema de produção é bastante modular: a base de dados, as regras e o controle são entidades independentes entre si;

(b) Como consequência do item acima, modificações na base de dados e alteração de regras não afetam o restante do sistema.

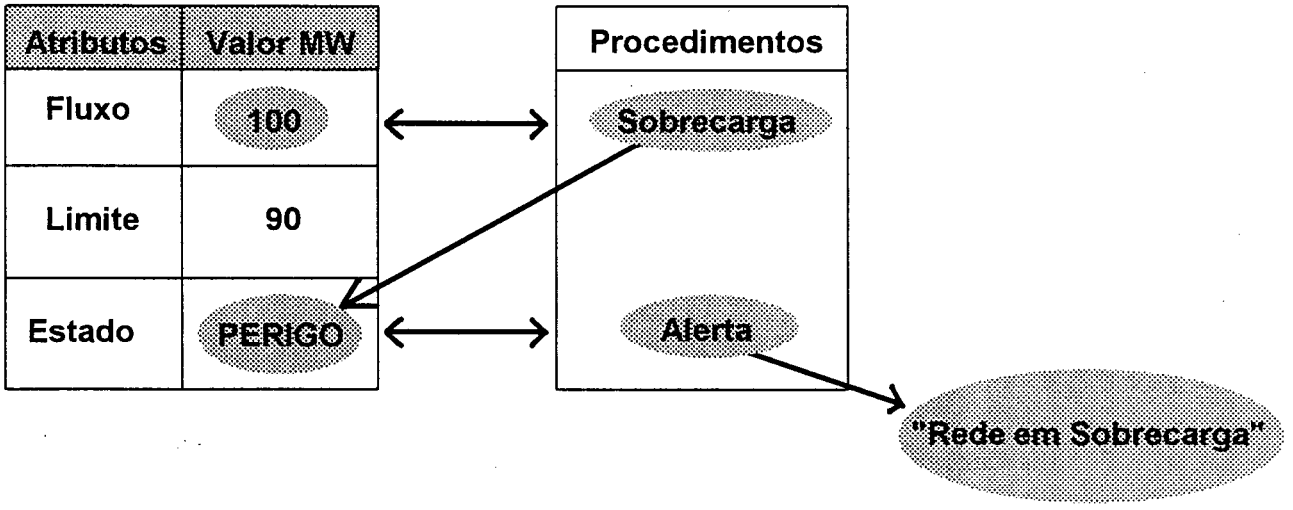
As regras de produção constituem um meio natural de descrever o comportamento de um problema que necessita mudar seus dados de entrada várias vezes (por exemplo, a configuração de um sistema elétrico de potência em um plano de expansão). As regras de produção providenciam um meio natural de representar recomendações, diretivas, ou estratégias adotadas por especialistas, sendo muito apropriadas quando o domínio do conhecimento resulta de associações empíricas desenvolvidas durante anos de experiência solucionando problemas de uma certa área.

4.2.2. Frames

A representação por frames e por redes semânticas pode ser vista como representação baseada em frames. Sua representação de conhecimento usa uma rede de nós conectados por relações e organizados dentro de uma hierarquia. Cada nó representa um conceito que pode ser

descrito por atributos e valores associados ao nó* . Cada atributo possui procedimentos associados ao mesmo, que são executados quando a informação (os valores dos atributos) é mudada. Nós abaixo da hierarquia herdam automaticamente os atributos dos nós de nível superior.

A melhor forma de compreender estes conceitos é através de um exemplo simplificado, mas que represente uma situação real, como consta na Figura 4.5.



Proc.Sobrecarga:
Se (Fluxo > Limit) Então Estado="PERIGO"
Senão Estado="NORMAL"

Proc.Alerta:
Se (Estado="PERIGO") Então EmitaMensagem("Rede em Sobrecarga")

FIGURA 4.5 - EXEMPLO DE REPRESENTAÇÃO DO CONHECIMENTO USANDO FRAMES

Neste exemplo, o usuário entra com um valor para o atributo *Fluxo*, no caso 100 kW* . Uma vez que o limite de carregamento sob condições normais para este circuito é 90 kW, o sistema está sobrecarregado. Tal verificação fica a cargo dos procedimentos *Sobrecarga* e *Alerta*,

---

* Exemplo:
Atributos: Nome, Número da Barra, Classe de Tensão
Valores associados: Farroupilha, 26, 230 kV

* O modo como o usuário passa este valor para a base de conhecimentos não é relevante para o presente estágio da apresentação. Apenas para citar, o usuário poderia passar este valor ou via teclado, ou através da leitura do arquivo de resultados de um programa que simulasse o fluxo de potência em uma rede qualquer.

que monitoram os valores dos atributos *Fluxo* e *Estado*, respectivamente, desencadeando a mensagem de alerta ao usuário.

Cada atributo pode ter tantos procedimentos quantos forem necessários. Há várias formas de associar um procedimento a um atributo, como mostrado a seguir [35]:

a) *Após mudar* (after change)

O procedimento é disparado após um novo valor ser passado ao atributo.

b) *Antes de mudar* (before change)

O procedimento é disparado imediatamente antes de um novo valor ser passado ao atributo.

c) *Quando acessado* (when accessed)

O procedimento é disparado quando o atributo for acessado e seu valor não for nulo.

d) *Se necessário* (if needed)

O procedimento é disparado sempre que for solicitado o valor do atributo e este for nulo.

Tais formas ditam como os procedimentos são disparados, sendo mais apropriadamente chamadas de monitores . Ditam também o chamado mecanismo de passagem de mensagens. Seria como se estes monitores passassem mensagens aos procedimentos, fazendo com que estes executassem as ações pertinentes. Tais ações podem incluir: passagem de mensagens a outros procedimentos da base de conhecimentos, mudança de valores nos atributos, execução de um conjunto de regras, etc.

Frames são uma forma mais flexível de representar o conhecimento, tornando-o modularizável ao permitir a definição de nós com conceitos gerais para os primeiros protótipos,

especializando-o com nós com conceitos mais singulares à medida que estes se façam necessários ao sistema.

Este tipo de representação é uma das formas mais claras de explicitar o conhecimento sobre uma dada área de estudo, uma vez que os nós hierarquicamente inferiores herdam os atributos e procedimentos associados aos nós de nível superior. Esta forma de representação do conhecimento permite criar protótipos simplificados, gerais, limitados, para posteriormente migrar para sistemas mais complexos e organizados.

Em sua evolução, este tipo de representação do conhecimento deu origem aos paradigmas de programação orientada a objeto [72], ou OOP (*Object Oriented Program*), que serão discutidos no capítulo a seguir (Capítulo 5). Fazendo um breve paralelo com o que foi exposto até agora, em OOP, os nós de nível superior são chamados de classes, das quais podem ser derivadas subclasses e/ou instâncias, estas últimas representando os nós de nível mais baixo. Os procedimentos presentes nos atributos são chamados de métodos e são acionados por mensagens, que funcionam como monitores. No entanto, a OOP traz consigo outras metáforas em sua formalização, como polimorfismo, encapsulação, herança múltipla, ausentes em SBC's.

4.3. CONCLUSÕES

O uso de técnicas de Inteligência Artificial faz com que o planejador disponha de um ferramental bem mais flexível para representar procedimentos heurísticos. Tais técnicas não visam substituir o homem por um sistema totalmente automático, e sim assisti-lo na condução dos seus estudos, de modo a obter uma melhoria na qualidade da solução gerada. A IA amplia a capacidade do planejador em propor novas soluções para o problema, pois o livra de tarefas repetitivas, triviais, podendo, assim, concentrar seus esforços, sua inteligência e sua imaginação na criação de melhores alternativas de expansão. De outro ponto de vista, o planejador estaria de

posse de um ferramental que não o deixaria negligenciar detalhes práticos básicos, uma vez que estes estariam incorporados ao modelo.

O presente capítulo procurou dar uma visão global do que vêm a ser sistemas baseados no conhecimento.

O próximo capítulo irá tratar dos paradigmas de programação orientada a objeto. Apesar de não ter sido a linha mestra de desenvolvimento do protótipo SAPIENS, tais paradigmas facilitam a compreensão da representação do conhecimento sob a forma de frames, sendo úteis em sua conceituação e aplicação.

CAPÍTULO 5 - PROGRAMAÇÃO ORIENTADA A OBJETO

5.1. INTRODUÇÃO

Para melhor compreensão da estrutura das linguagens de programação do protótipo objeto desta dissertação, este capítulo apresenta conceitualmente a filosofia de programação orientada a objeto [65] (OOP - *Object Oriented Programming*) , suas vantagens e singularidades.

O propósito fundamental da OOP é facilitar a modelagem de problemas complexos [64]. A chave para atingir essa meta é o conceito de *objeto*. A vantagem da utilização de objetos é que, quando corretamente aplicados, eles são mais fáceis de se compreender e manipular do que os elementos separados que constituem um projeto computacional.

5.2. PROGRAMAÇÃO ORIENTADA A OBJETO

A OOP representa o estágio mais recente no desenvolvimento da tecnologia de programação. É preciso entender que não se trata de uma linguagem propriamente dita, mas sim de uma filosofia de programação que se mostra atraente por causa de sua versatilidade em modelar conceitos genéricos.

Num estágio abaixo da OOP está a programação *estruturada*, que é o método utilizado por linguagens como C e Pascal. Empregando-se linguagens estruturadas, foi possível estudar problemas relativamente complexos de maneira razoavelmente fácil. Entretanto, mesmo com a programação estruturada, a capacidade de programação decresce com o aumento da complexidade dos problemas em estudo.

A OOP aproveitou as melhores idéias da programação estruturada e as combinou com novos conceitos que trazem mais luz à tarefa de programação. Permite assim, que um problema seja mais facilmente decomposto em subgrupos relacionados. Usando-se a linguagem adequada, pode-se traduzir tais subgrupos em unidades autocontidas chamadas objetos.

5.3. METÁFORAS DE PROGRAMAÇÃO ORIENTADA A OBJETO

A seguir, uma breve abordagem dos conceitos mais comuns presentes em linguagens OOP.

5.3.1. Objetos

A característica mais importante de uma linguagem OOP é justamente a definição de *objeto*. De maneira simples, objeto é uma entidade lógica que contém dados e códigos para manipular esses dados. Estes elementos podem ser privados ao objeto, ou seja, inacessíveis externamente. Dessa maneira, evita-se que outras partes não relacionadas de programa modifiquem ou usem incorretamente tais áreas privadas. Essa ligação dos códigos e dos dados é freqüentemente referenciada como *encapsulação*.

A encapsulação, por sua vez, nada mais é do que um adequado particionamento da memória em áreas exclusivas para dados e procedimentos, proporcionando uma independência funcional entre os objetos e o restante do programa.

Para todas as intenções e propósitos, um objeto é uma variável de um tipo definido pelo usuário. Pode parecer estranho à primeira vista encarar um objeto, que une tanto código como dados, como sendo uma variável. Contudo, em OOP, é precisamente esse o caso. Quando se define um objeto, cria-se implicitamente um novo tipo de dado.

5.3.2. Polimorfismo

Linguagens OOP suportam polimorfismo, o que significa essencialmente que um nome pode ser usado para muitos propósitos relacionados, mas ligeiramente diferentes. A intenção do polimorfismo é permitir o emprego de um nome para especificar uma classe geral de ações.

5.3.3. Classes, instanciação e herança

O conceito de classes é muito importante em OOP.

Uma classe é um gabarito para definir objetos reais. Uma classe é instanciada se ela tiver sido usada para estampar um ou mais objetos de seu tipo. Apropriadamente, os objetos que tiverem sido criados de uma determinada classe são chamados *instâncias* desta classe.

O que realmente dá justificativa prática para a estrutura de classe é a facilidade de *herança*. Herança é o processo pelo qual uma classe pode adquirir as propriedades de outra classe.

Por exemplo, quando uma **classe B** for definida com base em uma **classe A**, diremos que **B** é uma *subclasse* de **A**. Reciprocamente, também diremos que **A** é uma *superclasse* de **B**. Isso equivale a dizer que **B** *herda* todo o comportamento de **A** (Figura 5.1).

Ter uma classe herdando o comportamento de outra seria inútil se isso fosse tudo. A finalidade de criar uma subclasse é incluir algum comportamento adicional além do que é herdado. Em geral, uma subclasse é ajustada com um comportamento que lhe permite atuar como uma versão mais especializada de sua superclasse.

Alguns sistemas orientados a objeto também possuem o que é chamado de *herança múltipla*. Nesses sistemas, uma nova classe pode herdar de duas ou mais superclasses (Figura

5.1). De qualquer forma, o resultado é uma classe que pode estampar objetos com um comportamento mais especializado.

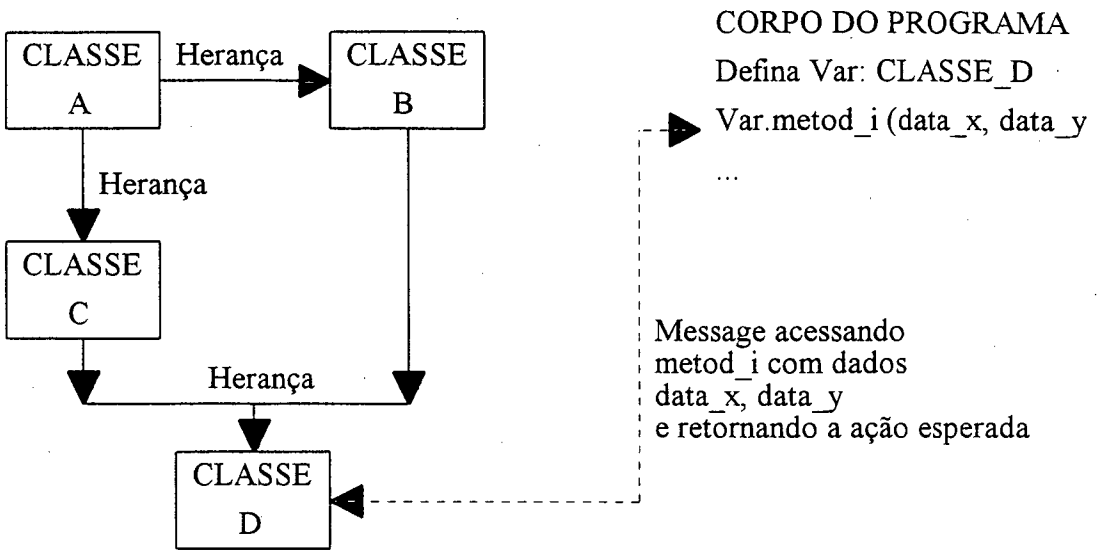


FIGURA 5.1 - CLASSES, DADOS ATIVOS E PASSAGEM DE MENSAGENS

Por meio desse processo de definição de subclasses, surge o conceito de *hierarquia de classes*, qual seja o de uma árvore ou rede de classes começando de elementos gerais, simplificados, para elementos mais específicos, detalhados.

Sem o uso de classificações, cada objeto precisaria definir explicitamente todas as suas características. Usando-se classificações, um objeto precisa definir somente aquelas qualidades que o tornam único dentro de sua classe, uma vez que ele herda as qualidades que compartilha com a classe mais geral.

Exemplificando, uma dada linha de transmissão faz parte da classificação *linha de transmissão*, que, por sua vez, faz parte da classificação *circuito*, que está na grande classe *sistema*.

Basicamente, uma classe representa a forma geral do objeto. Uma instância representa uma variável, que faz uso das definições presentes em uma dada classe. Novas classes podem ser

criadas a partir de uma classe base, herdando as características desta última e acrescentando algumas peculiariaridades úteis ao projeto (Figura 5.2).

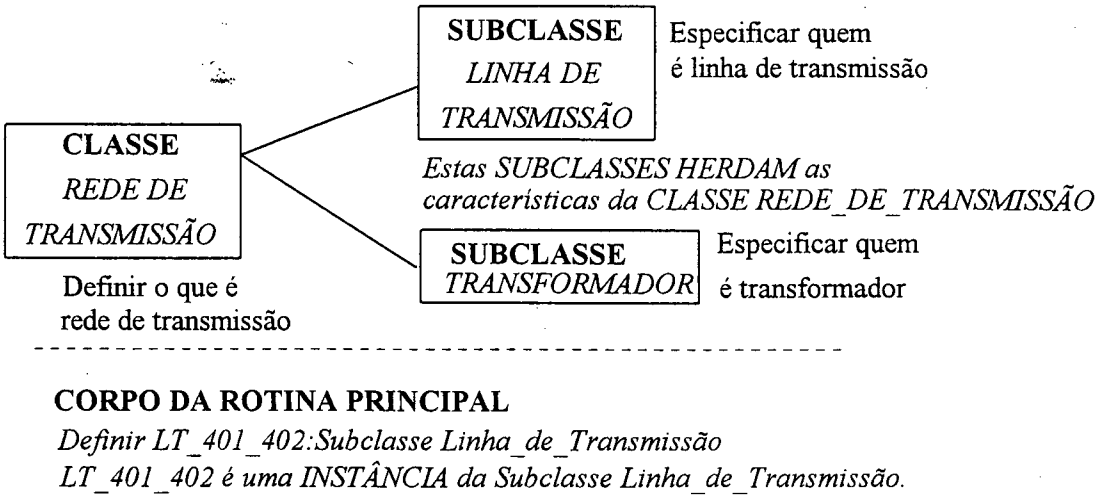


FIGURA 5.2 - CLASSES, HERANÇA E INSTÂNCIAS EM OOP

5.3.4. Passagem de mensagens e dados ativos

Algumas das metáforas mais conhecidas em OOP já foram citadas, quais sejam, *objeto*, *polimorfismo*, *herança*, *classe* e *instância*. Contudo, ainda é preciso esclarecer o significado real dado a duas outras metáforas, a saber, *passagem de mensagens* e *dados ativos*.

Uma das distinções mais básicas na programação é a de dados e subrotinas. A estrutura básica de um programa mostra que os dados são armazenados em chips de memória, enquanto as instruções dos processadores são usadas para manipular os dados. Além do mais, dados são essencialmente passivos e instruções são elementos ativos que operam sobre o elemento passivo.

A OOP procura definir uma nova estrutura para os programas, modularizando-os em classes, que suportam tanto códigos quanto dados. Os códigos, por sua vez, representam os métodos, as funções que a classe está apta a cumprir. No corpo do programa, após se instanciar

corretamente uma dada classe, a consecução das tarefas desejadas é promovida pela *passagem de mensagens*. Estas indicam, especificamente, quais dados e métodos serão utilizados (Figura 5.1).

A passagem de mensagens faz com que os objetos se comportem como se fossem dados ativos. Entretanto, no corpo de um objeto, tem-se a mesma dicotomia entre dados passivos e procedimentos ativos encontrada na programação convencional.

O modo como os objetos interagem entre si é cuidadosamente controlado pelo mecanismo de passagem de mensagens. Em geral, um objeto aceita mensagens que ele reconhece, e ignora aquelas que não reconhece. Como todas as interações seguem esse protocolo, não existe oportunidade do código externo a um objeto interferir com seu funcionamento de forma imprevisível ou indesejável.

No mínimo, enviar uma mensagem envolve especificar o nome de um objeto e o nome da mensagem a ser enviada a ele. Constantemente haverá *argumentos* que deverão ser especificados. Os argumentos podem ser nomes de *variáveis locais*, conhecidas apenas pelo tipo de objeto que reconhece a mensagem, ou podem ser *variáveis globais*, conhecidas por todos os objetos.

Nada obriga um sistema usar ou não a passagem de mensagens. No entanto, é fácil fazer uma breve analogia com a programação convencional. Nesta, as tarefas a serem cumpridas em um programa devem ser corretamente implementadas em subrotinas. Mesmo com avanços no sentido de se desenvolver sistemas computacionais com subrotinas compiladas à parte do corpo do programa, estas carregarão todo o código pré-definido. No caso de sistemas que usam a passagem de mensagens, e vendo-se os objetos como subrotinas gerais, é possível abster-se de algumas funções em seu bojo quando de sua chamada.

O mesmo não ocorre no caso de DLL's (*Dynamic Link Libraries*), rotinas pré-compiladas que se tornam disponíveis a qualquer programa executável, independente do compilador que o gerou. Tais rotinas, quando definidas, ficam à disposição de qualquer sistema que delas faça uso, sendo a estes ligadas em tempo de execução. Assim, somente o programa principal fica residente na memória, enquanto os objetos DLL, são carregados à medida que são requisitados (ver

Apêndice D). Este mecanismo é uma inovação trazida pelo ambiente Windows, a fim de promover a modularidade dos aplicativos gerados para o mesmo.

Na programação estruturada, a passagem de mensagens permite que técnicas avançadas de programação explorem a funcionalidade particionada dos objetos que respondem às mesmas. Embora existam meios de obter muitos dos mesmos resultados sem passar mensagens, geralmente isto é mais difícil na programação convencional* e exige maior conhecimento e habilidade por parte do programador. Em OOP, os objetos possuem um tipo muito especial de autonomia, inédita na programação. Em certo sentido, cada objeto se comporta como se fosse um pequeno computador especializado que se comunica com outros computadores especializados. Embora isso seja uma outra metáfora, o modelo de passagem de mensagens sugere muitos algoritmos interessantes e poderosos que podem ocorrer aos programadores e projetistas de software no sentido de contornar os limites da programação convencional.

5.4. Tipos de Sistemas Orientados a objetos

A diferença básica entre sistemas OOP pode ser escalonada em sistemas puros e híbridos. Um sistema puro é aquele em que todos os elementos são objetos. Neste tipo de sistema, até mesmo as classes são objetos que são instâncias de uma outra classe. A princípio isso pode parecer confuso, mas não existe nada de paradoxal a respeito. É, na realidade, um resultado direto do desejo de coerência total no sistema. Assim como os objetos são criados de uma classe que serve como seu modelo ou gabarito, as próprias classes são objetos criados de acordo com a planta estabelecida em uma classe específica. Em geral, isso é chamado metaclasse.

---

* Uma saída seria o uso de *flags*, variáveis que selecionassem a tarefa desejada quando setadas em um valor pré-definido.

Um sistema híbrido é aquele em que os objetos coexistem dentro de uma linguagem de programação convencional. Obviamente, por terem que coexistir com outras linguagens de programação, os elementos em sistemas híbridos podem não ser todos objetos.

Uma outra forma de classificação de sistemas OOP é pelo tipo de esquema de herança utilizado. Basicamente, existem dois tipos: herança única e herança múltipla. Os sistemas com herança múltipla são aqueles em que uma classe herda o comportamento de mais de uma superclasse. A herança múltipla ainda é uma característica relativamente nova em sistemas OOP e muitas implementações ainda não a possuem.

É preciso evitar que a herança múltipla se torne muito simplificada. Embora esta seja uma situação rara, ela deve ser manipulada de forma rigorosa a fim de evitar conflitos entre os nomes de variáveis e os nomes em diferentes classes que devam ser herdadas. Até o momento ainda não existe um método-padrão de manipulá-la, com o qual todos concordem [65].

Uma distinção final dos diferentes tipos de sistemas OOP é entre aqueles que possuem um processamento concorrente real e aqueles que não o possuem. Em um sistema de passagem de mensagem concorrente, por exemplo, as mensagens que são trocadas entre os objetos podem ocorrer em paralelo, em vez de esperar por sua vez, como nos sistemas de processamento sequencial.

A seguir, o Quadro 5.1, lista cinco critérios que medem o grau pelo qual um sistema é orientado a objetos [65].

QUADRO 5.1 - GRAUS DE SISTEMAS ORIENTADOS A OBJETO

1.	Classes com múltiplas intâncias
2.	Funções ou dados encapsulados
3.	Ligação run-time
4.	Herança múltipla
5.	Combinação de mensagens

5.5. CONCLUSÕES

Existem vários motivos para que os programadores achem o método orientado a objetos importante e até mesmo indispensável. Para um programador, poderia ser mais atrativo a possibilidade de eliminar códigos redundantes. Para outro, poderia ser a capacidade de proteção que os objetos têm de serem invadidos por códigos de outras partes do programa. Para outros ainda, poderia ser a economia de tempo aliada à possibilidade de construir programas a partir de partes de trabalho padronizadas, que se comunicam entre si, em vez de ter que começar a escrever o código do nada. Finalmente, existem programadores que gostam mais da capacidade de ter muitas instâncias de um objeto presentes simultaneamente sem quaisquer interferências.

A programação orientada a objetos não é uma direção inteiramente nova e sem precedentes na ciência de computação. Ela simplesmente leva os desenvolvimentos recentes das técnicas de inteligência artificial para o próximo passo lógico, ou seja, maior clareza, modularidade e eficiência de programação.

Há pelo menos três vantagens imediatas nos sistemas orientados a objetos. Primeiramente, é possível ter várias instâncias de uma dada classe presentes no sistema ao mesmo tempo, enquanto a memória assim o permitir. Uma classe é simplesmente um gabarito em que cada objeto é

modelado, recebendo sua própria área de memória que não pode ser acessada por qualquer outro objeto, a não ser usando os próprios métodos locais do objeto.

A segunda vantagem é que, por meio de um mecanismo de herança, subclasses compartilham as variáveis e métodos de suas superclasses. Isto é, maiores especializações de funções podem ser escritas incluindo-se apenas a parte única, enquanto o restante é herdado automaticamente.

A terceira vantagem imediata é a de poder fornecer uma interface uniforme sobre a faixa mais ampla possível de tipos de objeto. O mesmo nome pode ser usado para métodos que devam ser implementados de forma diversa em variados objetos, enquanto as diferenças permanecem invisíveis ao usuário.

Alguns dizem que a vantagem principal da OOP é a capacidade de reutilizar o código para muitos programas diferentes. Mas, por si só, isso não é muito diferente das funções de biblioteca. Uma diferença mais substancial é a capacidade melhorada para cuidar deste problema de forma transparente.

Uma das principais vantagens da OOP é seu poder em lidar com problemas que geram programas convencionais muito extensos. No entanto, essa peculiaridade não ocorre automaticamente. Dividir um programa extenso em partes consistentes é uma parte do processo. É igualmente necessário aprender a utilizar as técnicas corretas para gerenciar o código e facilitar o trabalho para os membros de uma equipe de programação. Os sistemas orientados a objetos são normalmente dispersos, com partes de aplicações sendo espalhadas entre diversas classes e subclasses. Para programar de forma eficiente com sistemas deste tipo, é essencial ter as ferramentas apropriadas e uma metodologia eficiente para manter a aplicação bem focalizada e organizada.

CAPÍTULO 6 - MODELAGEM DO PROTÓTIPO SAPIENS

6.1. INTRODUÇÃO

Nos capítulos anteriores deste trabalho, procurou-se definir os aspectos conceituais pertinentes ao planejamento de sistemas de potência (vide Capítulo 2), inteligência artificial (vide Capítulo 3), sistemas baseados no conhecimento (vide Capítulo 4) e programação orientada a objeto (vide Capítulo 5). Tendo por base estes conceitos, o presente capítulo descreverá a metodologia empregada na implementação do protótipo SAPIENS (*Sistema Adaptativo e Interativo de Planejamento da Expansão*), um sistema baseado no conhecimento de apoio a decisão ao processo de planejamento*.

6.2. EVOLUÇÃO DE UM SISTEMA BASEADO NO CONHECIMENTO

A Figura 6.1 descreve as etapas básicas de criação e aprimoramento de um sistema baseado no conhecimento [72], conforme será discutido a seguir.

Após completar a fase inicial de revisão bibliográfica sobre um dado tema a modelar, o próximo passo é a criação de um protótipo, ou seja, um modelo experimental. Para facilitar a modelagem e implementação do protótipo, o escopo do problema deve ser reduzido ao mínimo. Por ser uma fase inicial, os propósitos a que ela se destina precisam ser bem consistentes, buscando situações que se possam comprovar com uma modelagem simplificada. Além do mais, tais propósitos ajudam a definir linhas de desenvolvimento para o sistema final em sua versão

* Uma vez que os paradigmas de sistemas baseados no conhecimento se assemelham aos de programação orientada a objeto, no decorrer desta dissertação serão usados indistintamente.

comercial. Assim, quanto à sua aplicabilidade, tais protótipos acabam por se mostrar lentos e ineficientes, consumindo mais tempo computacional do que o necessário. Além disso, o suporte técnico aos seus usuários (muitas das vezes, a própria equipe que o está implementando) é visto como sendo um objetivo secundário nesta fase de desenvolvimento.

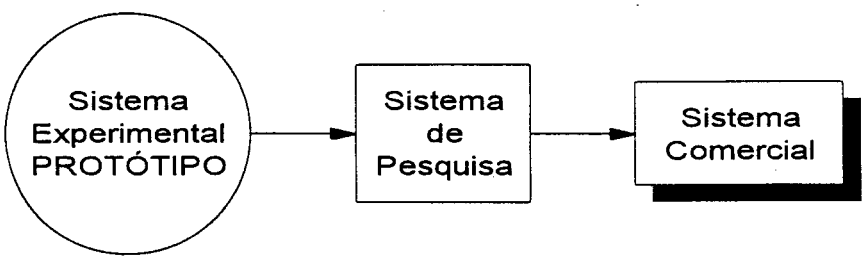


FIGURA 6.1 - EVOLUÇÃO DE UM SISTEMA BASEADO NO CONHECIMENTO

O protótipo SAPIENS, por exemplo, foi modelado para lidar apenas com linhas de transmissão e transformadores, fazendo análise em regime permanente através da simulação da rede em estudo com rotinas de fluxo de potência DC (modelo simplificado e não iterativo).

As versões iniciais do protótipo foram desenvolvidas em poucas semanas. A versão atual foi implementada com pouco mais de seis meses de trabalho, contando apenas com duas pessoas na equipe de desenvolvimento. A nível de aplicabilidade, apesar de ser mais lento do que modelos correlatos (SINTRA, TANIA, etc.), o protótipo SAPIENS é mais interativo do que esses, como se verá nos capítulos que seguem. Sua lentidão advém do seu *motor de simulação* (rotina responsável pelo processamento numérico), escrito em FORTRAN 77 LAHEY, uma linguagem de programação para DOS, enquanto o protótipo foi escrito para operar em ambiente WINDOWS*.

Outros tipos de circuitos poderiam ter sido considerados na modelagem do protótipo, bem como modelos de simulação mais elaborados. Contudo, mostrou-se mais eficaz, do ponto de vista de implementação, tratar o protótipo da forma como foi organizado, isto é, com uma base de

* Para fazer análise do sistema em regime permanente, o protótipo SAPIENS necessita abrir um *shell* para DOS, isto é, uma área de trabalho na memória disponível onde é executada a rotina de fluxo de potência DC. Toda vez que um programa compilado em DOS for executado em ambiente WINDOWS, isto irá acontecer.

conhecimentos rudimentar e rotinas de simulação simplificadas, mas que retratavam adequadamente o processo de planejamento da expansão de redes de transmissão.

Seguindo a Figura 6.1, um sistema de pesquisa seria uma versão melhorada de um sistema experimental. O uso adequado das técnicas de inteligência artificial, sistemas baseados no conhecimento e programação orientada a objeto, ao proporem uma nítida separação entre base de conhecimentos, cadeias de métodos e rotinas de simulação, permitem que este melhoramento ocorra como uma especialização do protótipo corrente. A nova versão assim constituída seria testada a fim de se comprovar se as particularidades introduzidas no modelo original fariam com que o mesmo suportasse questões mais generalizadas, deixando de se limitar a propósitos específicos. Com a experiência adquirida ao longo do seu desenvolvimento, poderia ser dado suporte técnico às equipes de testes, interagindo com as mesmas na forma de entrevistas, nas quais seriam levantados pontos críticos, que, quando superados, melhorariam a aplicabilidade do modelo (estudos de ergonomia). Por isso, mesmo sendo um modelo mais elaborado, um sistema de pesquisa ainda pode se apresentar como sendo um sistema lento e ineficiente, carecendo de melhoramentos que o tornem mais prático e abrangente.

Por fim, um sistema comercial seria um modelo validado, pronto a atender seus usuários dentro de sua área de atuação. Contaria com extenso material para compor uma boa equipe de suporte e se apresentaria como um sistema rápido e eficiente. Para atingir esta fase, os esforços devem se concentrar no refinamento do sistema de pesquisa, ou parte deste que se apresente como modelo útil a nível comercial. Não são tolerados enganos, ou erros de precisão. Os resultados deste último modelo devem ser consistentes e garantir um rico volume de informações aos seus usuários.

A nível coloquial, um sistema experimental seria similar a uma *criança*, moldando-se em um mundo desconhecido e pleno de experiências. Um sistema de pesquisa seria um *adolescente*, uma pessoa consciente, mas que ainda nutre certas dúvidas. Sob esta ótica, um sistema comercial poderia se assemelhar a um *adulto*, uma pessoa consciente e capaz de assumir suas responsabilidades, mesmo não sendo o *dono da verdade*.

Tal esquema serve para demonstrar a utilidade da redução do escopo de estudo conforme o nível de detalhamento que se queira em modelos computacionais. Esta redução do escopo de estudo só se consegue ao estipular adequadamente os objetivos almejados. Mediante entrevistas com especialistas da área e pesquisas afins, tais objetivos vão se tornando mais claros. Não raro, ocorrem objetivos intermediários, que fogem do escopo do problema original, mas são necessários ao trabalho de implementação.

Como exemplo, no desenvolvimento do protótipo SAPIENS, cerca de dois meses foram dedicados a pesquisas por rotinas de simulação e técnicas de programação que gerassem aplicativos adequados ao ambiente WINDOWS.

O sucesso em se conseguir um produto final, de valor prático, está na adequada aplicação da metodologia exposta a seguir.

6.3. METODOLOGIA EMPREGADA

A justificativa para criação de um novo modelo de síntese se prende às deficiências presentes na atual metodologia de planejamento da expansão a longo prazo de sistemas de transmissão, no que se refere às formas de representação e manipulação das heurísticas de planejamento para a síntese automática de redes de transmissão, a saber:

a) Não há interação entre o planejador e o computador; o computador não explica situações existentes, nem aprende com situações novas;

b) O planejador não tem acesso às heurísticas; a modificação ou inclusão de qualquer heurística é muito trabalhosa, exigindo do usuário um bom conhecimento do código do programa;

c) A fixação dos procedimentos heurísticos no próprio programa é desvantajosa. Devido às peculiaridades de cada rede elétrica, um conjunto de regras ótimas para o

planejamento de determinados sistemas de transmissão pode não ser ideal para outros.

Foi principalmente pela necessidade de um trato mais refinado das heurísticas de planejamento que se optou pelo desenvolvimento de um novo modelo de síntese, com base em técnicas de sistemas baseados no conhecimento. A preocupação com a possibilidade de registro do conhecimento técnico dos especialistas na área, somou-se a este fator na motivação para o desenvolvimento de um protótipo que desse apoio ao processo de planejamento da transmissão.

Como cita [72], a Figura 6.2 apresenta uma metodologia consistente para desenvolvimento de um sistema baseado no conhecimento. Esta metodologia pode ser empregada em qualquer uma das etapas de desenvolvimento entre um protótipo e um sistema comercial.

a) IDENTIFICAÇÃO

Identificação e interpretação do problema pela equipe de desenvolvimento, indicando requisitos de hardware, metas e objetivos a alcançar. É a fase onde se define o escopo do problema.

b) CONCEITUAÇÃO

Fase em que engenheiros de conhecimento, responsáveis pela aquisição do conhecimento, junto com os especialistas, definem os conceitos, relações e mecanismos de controle necessários para modelar e simular o fluxo de solução do problema.

c) FORMALIZAÇÃO

Expressa os conceitos e relações chaves sob a forma de uma base de conhecimentos, seja esta esquematizada em fatos e regras de produção, frames, redes semânticas ou classes de objetos. Define-se qual a melhor forma de representação dentro do escopo do problema, bem como as ferramentas de desenvolvimento.

d) IMPLEMENTAÇÃO

Codificação do programa computacional. O sistema precisa ter conteúdo, forma e integridade para funcionar bem. O conteúdo foi estabelecido pela FORMALIZAÇÃO, isto é, estrutura de dados, regras para inferência e estratégias de controle. A forma é dada pela linguagem de programação. A integração visa combinar e reorganizar as várias partes do conhecimento para evitar que ocorram enganos no fluxo de execução.

e) TESTES

Avaliação do desempenho e utilidade do protótipo, ocasionando revisões nas versões implementadas através de reformulações, redefinições e refinamentos no código fonte. As seguintes questões influem neste sentido:

i. O sistema toma decisões aceitas pelos especialistas?

ii. As regras de inferência estão corretas, consistentes, completas?

iii. A estratégia de controle do sistema permite considerar seus elementos constituintes na mesma ordem adotada pelos especialistas?

iv. O sistema explica bem o porquê de suas ações?

v. O sistema cobre o domínio proposto?

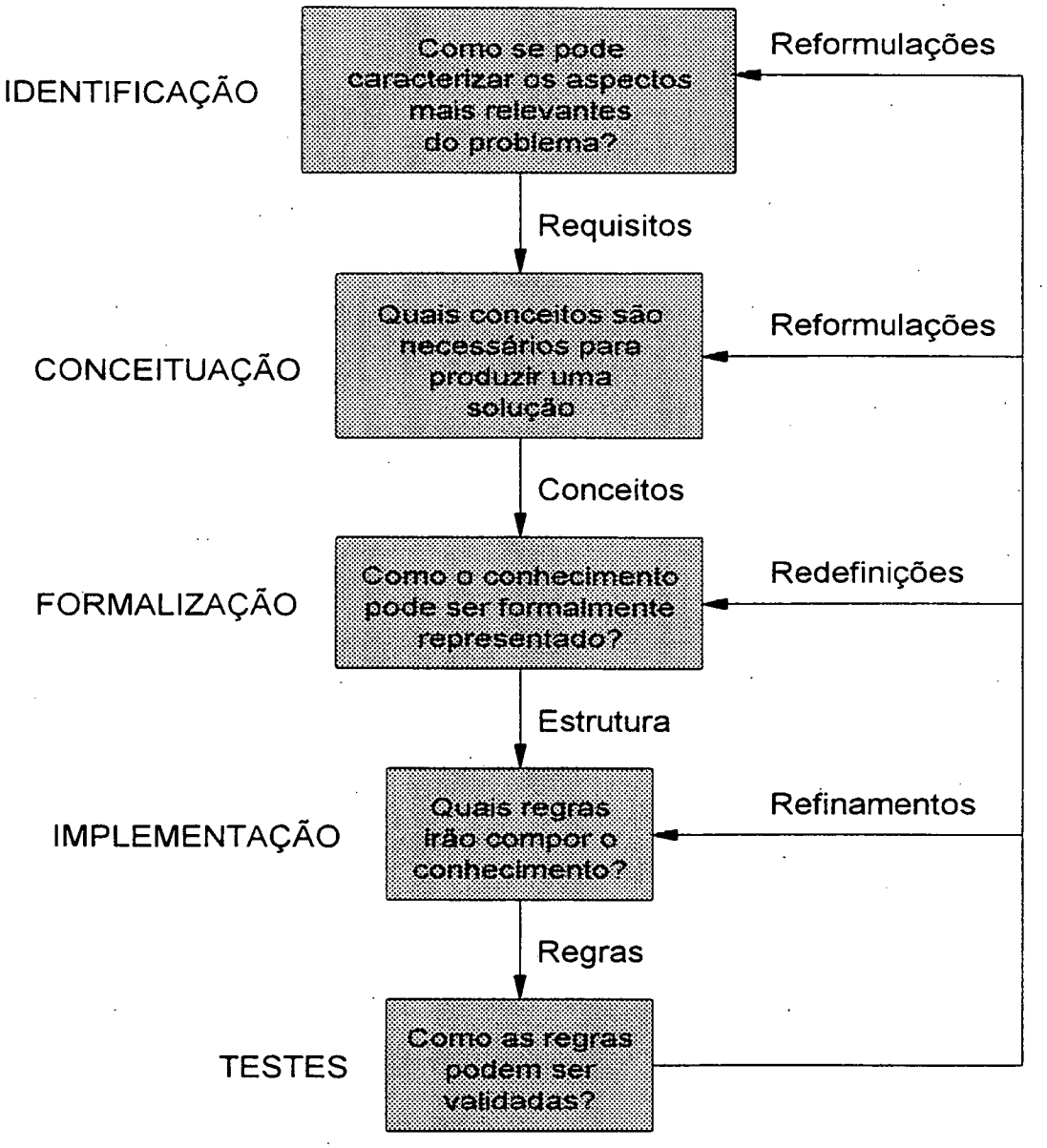


FIGURA 6.2 - FASES DE DESENVOLVIMENTO DE UM SISTEMA BASEADO NO CONHECIMENTO

6.3.1. Aplicação da metodologia proposta

Como exemplo de aplicação da metodologia proposta, citam-se os passos dedicados no desenvolvimento do protótipo SAPIENS, esboçados na Figura 6.3 e discutidos a seguir.

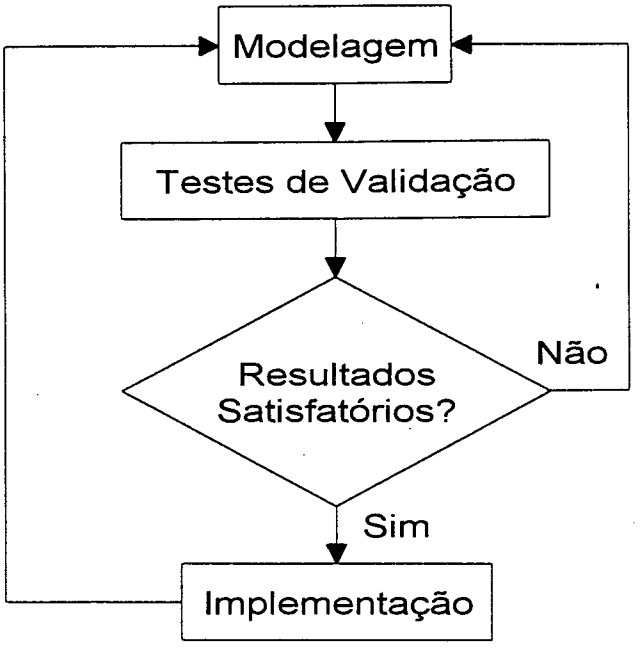


FIGURA 6.3 - PASSOS PARA DESENVOLVIMENTO DO PROTÓTIPO

a) Modelagem

Esta fase define a estrutura da base de conhecimentos e as rotinas de simulação.

Para o protótipo em questão, a base de conhecimentos foi representada na forma de *frames*, na verdade, classes de objetos com atributos e métodos associados. A melhor estratégia encontrada foi partir do mais geral para o mais simples, especializando-se à medida que o conhecimento relacionado ao problema em estudo fosse se descortinando ao longo das pesquisas. O acesso a este conhecimento se deu mediante entrevistas com especialistas da área (vide Apêndice B), e também com estudos de planejamento realizados com o programa SINTRA.

As rotinas de simulação adotadas são aquelas presentes no modelo SINTRA, versão de Set/92. A escolha destas rotinas se deu por dois motivos. Primeiro, porque o modelo SINTRA é um modelo validado e aceito pela ELETROBRÁS para estudos de síntese de redes de transmissão em planejamento a longo prazo. Segundo, pela

possibilidade de acesso ao seu código fonte, o que viabilizou as modificações necessárias ao presente trabalho. Dentre estas, a principal modificação realizada foi fazer com que o modelo SINTRA pudesse ler seus comandos a partir de um arquivo de dados previamente especificado.

Nesta fase, também foram definidas funções simplificadas de interesse ao protótipo. Tais funções são responsáveis pela interação com o usuário, passando-lhe avisos, alertas e diagnósticos simplificados da rede em estudo. Tais funções foram responsáveis pela formação de uma interface gráfica onde é plotado o diagrama unifilar da rede em estudo. A escolha deste tipo de interface adveio da dificuldade sentida na experiência com trabalhos onde arquivos textos eram a única fonte possível quando se queria visualizar a configuração da rede em expansão.

b) Testes de Validação

É a fase de *simulação* das heurísticas implementadas. Através de valores aleatórios nos atributos dos objetos da base de conhecimentos, são disparados métodos associados e observado o comportamento das funções definidas pelo usuário. Para tanto, é comum o uso de *flags*, sinalizadores que indicam o estado do protótipo, caracterizam sua base de conhecimentos e o cenário corrente do sistema em estudo.

É o momento de ajuste do protótipo em busca do comportamento adequado de suas partes constituintes. Mesmo a interface com o usuário, ou a completa redefinição da base de conhecimentos pode ser definida nesta etapa, havendo um desvio no fluxo para a primeira etapa como se vê na Figura 6.3.

No caso do protótipo SAPIENS, as primeiras versões pouco se diferenciaram. A última versão é mais estruturada e sua base de conhecimentos mais especializada (vide Apêndice C).

Para melhor controle do processo de desenvolvimento é conveniente manter cópias das versões anteriores e passar a um teste por vez. Quanto mais especializado for um protótipo, mais complexo será seu mecanismo de controle (encadeamento dos métodos e chamadas de funções). Por ser um sistema baseado no conhecimento, sua sequência de execução não segue um comando por vez, como se vê em programas convencionais. O fluxo de dados fica interrompido até que algum comando seja passado (alguma mensagem seja passada a algum método, ou algum valor de certo atributo seja solicitado).

A adequada documentação dos passos dados serve como indicador para um possível retorno aos passos iniciais. Muitas vezes, em decorrência deste fato, é mais útil retornar à versão anterior, descartando as mudanças atuais, do que insistir em novos testes e arcar com a possível perda de continuidade do projeto.

c) Implementação

O último passo é meramente elucidativo. É neste passo que ocorre o refinamento das mudanças feitas na primeira etapa e que foram validadas na segunda etapa. O protótipo volta ao estado inicial, com os atributos dos objetos da base de conhecimentos em seu estado *default*.

Esta é a versão entregue para efeito de estudos de casos, sendo muito úteis relatórios de atividades sobre as seções de estudo. Tais relatórios são úteis no sentido de poder ditar limitações do protótipo, além de assinalar pontos que seriam dúbios a usuários.

6.4. ARQUITETURA DO PROTÓTIPO

O presente protótipo assenta-se sobre o ambiente Windows, ambiente gráfico multitarefa, tendo sido implementado utilizando rotinas em C ANSI, e FORTRAN 77 (vide Apêndice A, C e

D). A ferramenta de suporte para prototipagem do sistema baseado no conhecimento foi o *shell* KAPPA-PC, da Intellicorp Inc. Para análise de redes de transmissão em regime permanente foram empregadas as rotinas presentes no modelo SINTRA (vide Capítulo 2 e Apêndice A).

A seguir, uma breve discussão sobre os passos dados.

6.4.1. Ambiente de trabalho

O ambiente Windows oferece vários benefícios para quem lida com software [55].

6.4.1.1. Benefícios do Windows para usuários convencionais

a) alta padronização entre os aplicativos presentes no mercado;

b) não há necessidade de configurar os periféricos* para cada aplicação. O Windows providencia *drivers* padrões de configuração para a maioria dos periféricos conhecidos;

c) cooperação e comunicação interprocessos (DDE - dynamic data exchange; OLE - object link exchange; etc.);

d) ambiente multitarefa, o que propicia execução de várias aplicações ao mesmo tempo;

e) gerenciamento de memória. O ambiente Windows suporta modo protegido nos microprocessadores 80286, 80386 e i486, e suporta memória virtual nos 80386 e i486.

---

* Vídeo, impressora, teclado, etc.

6.4.1.2. Benefícios do Windows para programadores

a) dispositivos independentes de gráficos. Assim, aplicações gráficas funcionam em todos padrões de adaptador de vídeo;

b) suporte imediato para uma grande faixa de impressoras, monitores e dispositivos apontadores, tais como mouse e trackballs;

c) uma vasta biblioteca de rotinas gráficas;

d) mais memória para grandes programas;

e) suporte para menus, icons, bitmaps, e muito mais.

Em vista destes benefícios, optou-se pelo Windows como ambiente para conter o protótipo SAPIENS, favorecendo principalmente o usuário final que passaria a contar com um modelo de síntese em ambiente multitarefa.

6.4.2. Ferramenta de desenvolvimento

A Figura 6.4 [72] ressalta que o nível de integração de um projeto depende da ferramenta de suporte utilizada em seu desenvolvimento.

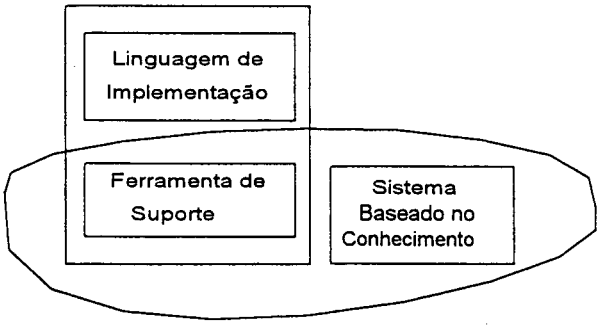


FIGURA 6.4 - FERRAMENTA PARA DESENVOLVER SISTEMAS DE CONHECIMENTOS

A arquitetura de sistemas baseados no conhecimento está vinculada à ferramenta de suporte adotada. A *Intellicorp Inc.* desenvolveu um pacote para desenvolvimento de protótipos de sistemas baseados no conhecimento, disponível ao GPSE (Grupo de Planejamento de Sistemas de Energia Elétrica) - UFSC (Universidade Federal de Santa Catarina) para estudos e análise. Trata-se do KAPPA-PC, versão 2.0 ([33], [34] e [35]).

Na verdade, o KAPPA é uma linguagem interpretada, sendo útil na prototipagem de sistemas experimentais. Um dos principais motivos que levou à adoção deste *shell*, foi sua facilidade em gerar código fonte em linguagem C, que pode ser posteriormente transformado em código executável com o emprego de um compilador adequado.

Os problemas de interface entre linguagens eram cruciais no passado. Tais problemas foram minorados com este pacote devido à sua biblioteca de funções pré-definidas. A possibilidade de gerar programas executáveis no formato DLL (*Dynamic Link Library*), abre as portas para ambientes computacionais onde módulos gerados em bases diferentes de software (FORTRAN, C, etc.) podem agora permutar seus dados (valores de variáveis, vetores e matrizes) como se fossem um único código. Tal ambiente se alia às facilidades proporcionadas pelo W*indows*, ambiente interativo e amigável.

6.4.3. Motor de simulação

Ao longo do desenvolvimento do protótipo SAPIENS, logo se notou a necessidade de um programa confiável para simulação de fluxo de carga e análise de contingências.

Como foi visto no Capítulo 2, existem dois modelos conhecidos para simular o planejamento da expansão de redes de transmissão de energia elétrica: o SINTRA e o TANIA. Enquanto o primeiro permite certa interação homem-máquina, o mesmo não ocorre com o segundo, um modelo totalmente automatizado. Ambos são modelos convencionais, baseados em rígidos algoritmos.

Uma vez que o SINTRA é um modelo validado pelo setor elétrico, optou-se pelo mesmo a fim de constituir o *motor de simulação* do protótipo SAPIENS.

Desenvolvido em linguagem FORTRAN 77, a filosofia adotada por seus programadores foi a de subdividir as tarefas ao máximo possível, padronizando-as e escalonando-as em blocos interdependentes (uso de recursividade). Esta é uma excelente metodologia de programação, estando a definição de variáveis separada dos procedimentos que as empregam*.

Algumas alterações no modelo SINTRA foram necessárias para que ele se adequasse ao protótipo SAPIENS, deixando de ser um modelo de apoio decisório e passando a ser um modelo de simulação. Tais alterações ocorreram graças à concessão, por parte do CEPEL, dos arquivos fontes do modelo SINTRA. Para tanto, um trabalho de análise se fez para identificar os pontos de ruptura do modelo SINTRA entre suas partes lógicas e algorítmicas.

Dois meses foram dedicados a pesquisas em busca de um compilador adequado ao modelo SINTRA. As DLL's implementadas no SAPIENS foram escritas em linguagem C ANSI (vide Apêndice D) e estavam voltadas para o ambiente Windows. O modelo SINTRA foi escrito em FORTRAN 77. A primeira idéia foi procurar por um compilador FORTRAN que gerasse DLL's. Foram encontrados dois, o *Fortran 5.1 da Microsoft Inc.* e o *WATCOM Fortran*. A ausência de manuais resultou em dois meses de busca ansiosa por um módulo DLL que executasse o fluxo de potência.

Mesmo simplificado, o SINTRA ainda se apresentava como um modelo extenso. Não seria fácil traduzi-lo para linguagem C. Buscou-se, então, um modelo mais simplificado, mesmo que não fosse tão completo quanto o SINTRA. Este modelo, foi o que desenvolvi durante meu período de créditos das cadeiras oferecidas pelo Mestrado. Trata-se de um modelo de fluxo DC que tomei o cuidado de escrever de forma estruturada.

---

* Contudo, cabe aqui uma palavra de alerta, o estreito interrelacionamento entre seus blocos constituintes, torna complexa sua manutenção, ficando esta limitada aos técnicos que o desenvolveram. Qualquer *toque* no código fonte deve ser dado de forma *cirúrgica* e com criterioso monitoramento sobre um caso base conhecido. A inobservância desta regra pode custar a invalidação completa do modelo e a perda do sentido de fluxo de dados.

Contudo, a tradução de um programa Fortran para linguagem C não é uma tarefa trivial. As versões atuais da linguagem C ([5], [55], [64], [69]) trazem consigo, em suas bibliotecas, a maioria das funções oferecidas pelo Fortran 77, este visto como uma linguagem científica, para cálculos numéricos. O tratamento de arquivos, variáveis, passagem de parâmetros nas chamadas de subrotinas, e, principalmente, o tratamento de vetores e matrizes, não é o mesmo nestas duas linguagens. A linguagem C, por exemplo, começa a indexar seus vetores e matrizes a partir do índice 0 (zero) enquanto Fortran inicia pelo índice 1 (um). O fluxo lógico do modelo utiliza variáveis temporárias para fazer variar estes índices, sendo que seus valores iniciais, e os *loops* que as manipulam não são exatamente os mesmos quando vistos de outra linguagem. O resultado final, mesmo com este modelo simplificado, após um mês de trabalho, foi um programa impreciso em seus resultados, *uma caixa preta deturpada*.

Novos caminhos foram embrenhados. Encontrou-se então um tradutor de Fortran para C produzido pela AT&T Bell Laboratories [20], um programa de livre circulação adquirido através da utilização da *internet*[*] .

Todavia, este pacote está voltado para UNIX, mas a AT&T permite que seja usado seu código fonte para gerar tradutores dedicados. O principal problema está ligado às suas bibliotecas de tradução de funções do Fortran, aglomeradas em dois pacotes de bibliotecas para funções I/O e funções que não são I/O. O pacote vem com um *makefile* com diretivas UNIX de compilação para gerar arquivos *.LIB* (MSDOS) ou *.a* (UNIX), bem como os headers *.h*.

O GRUCAD (Grupo de Concepção e Análise de Dispositivos Eletromagnéticos), laboratório da UFSC, também esteve interessado neste tradutor, tanto para UNIX quanto para MSDOS. Chegou a desenvolver um tradutor para MSDOS, com base nos arquivos disponíveis. Mesmo assim, não foi possível traduzir o programa de fluxo de potência para um código fonte em C que pudesse ser compilado para Windows.

---

[*] E-mail *netlib@research.att.com* no subdiretório */netlib/f2c* ou */netlib/f2c/msdos* com a extensão *.Z* (sigla de compactação para estações de trabalho).

Para que o trabalho não ficasse parado, decidiu-se seguir em frente, sem compilar de forma diferente as rotinas numéricas do programa SINTRA. Estas permaneceram com seus códigos fontes como são usados pelo MSDOS

Esperava-se subdividir o modelo SINTRA em seus módulos básicos de fluxo de potência, estimação de índices de sensibilidade e análise de contingência. O custo de desenvolvimento logo se viu ser maior do que o esperado, ficando como hoje está.

Em sua versão modificada, o modelo SINTRA não mais interage com o usuário, interpretando os comandos usuais de síntese, análise e gerais [7] a partir de um arquivo de dados que também contém os nomes dos arquivos do caso base e do arquivo de adições. Este arquivo de comandos é gerado pelo protótipo SAPIENS.

6.4.4. Base de conhecimentos

A base de conhecimentos traduz o nível de compreensão do protótipo dentro da sua área de abrangência. No presente caso, diz como o protótipo consegue representar e lidar com uma rede de transmissão de energia elétrica.

Dentro da nomenclatura usual de sistemas baseados no conhecimento, o protótipo SAPIENS está organizado na forma de *frames*. Na verdade, sua configuração atende aos paradigmas de programação orientada a objeto. Sua base de fatos contém métodos associados aos atributos de suas classes de objetos. Tais métodos disparam e controlam o fluxo do conhecimento em uma seção de estudo.

A Figura 6.5 descreve a última atualização da base de conhecimentos do protótipo SAPIENS.

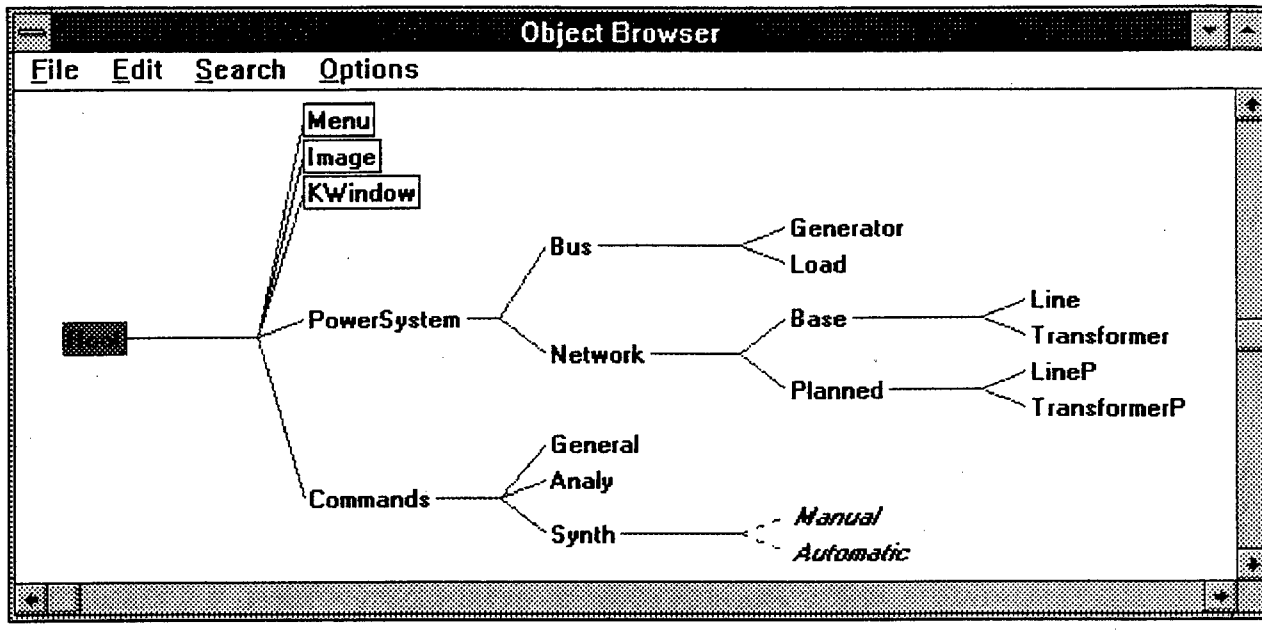


FIGURA 6.5 - BASE DE CONHECIMENTOS SOB A FORMA DE FRAMES

Duas instâncias em especial, *Default* e *Study*, contêm atributos gerais para a seção de estudo. Tais atributos visam caracterizar o estado em que se encontra o trabalho. Por exemplo, quando o atributo *Start* contém o valor ON indica terem sido definidos os arquivos de dados, habilitando os menus e *Buttons* de comando, bem como abrindo a janela com apresentação gráfica da rede em estudo.

Os fatos relevantes, úteis ao planejamento da expansão de sistemas de transmissão, que estão incluidos na base de conhecimentos do protótipo SAPIENS, são:

a) Dados de barras - numeração da barra, nome, classe de tensão, geração [MW], carga [MW], localização;

b) Dados de ramos (LT's e trafos) existentes - numeração da barra inicial, numeração da barra final, tipo do ramo (LT / trafo), capacidade normal de carregamento;

c) Dados de ramos planejados - numeração da barra inicial, numeração da barra final, tipo (LT / trafo), capacidade normal de carregamento, faixa de passagem, número de circuitos existentes.

Além destes dados, a base de conhecimentos tém representadas algumas heurísticas de planejamento, quais sejam:

a) Métodos para monitorar os limites de carregamento dos ramos de expansão, em condições normais;

b) Métodos para monitorar os limites das faixas de passagem;

c) Métodos para verificar existência de barras isoladas com injeção de potência ativa diferente de zero.

Existem ainda outros métodos para tarefas mais específicas, úteis na solução de problemas diferenciados de planejamento da expansão de sistemas de transmissão, tais como métodos para criação de caminhos de reforço para a rede elétrica. Contudo, tais métodos não foram implementados na versão atual do protótipo.

Por exemplo, a Figura 6.5 descreve a estrutura em frames de uma base de conhecimentos de interesse para o planejamento de redes de transmissão. Cada nó nesta base de conhecimentos expressa um conceito específico, partindo do mais geral, a descrição do que vem a ser um sistema de potência (*Power System*), para níveis mais específicos, contendo os vários tipos de circuitos que o completam (*Line, LineP, Transformer* e *TransformerP*). Os nós de nível intermediário são subconceitos, úteis para explicitar o conhecimento modelado (*Network* - rede; *Base* - circuitos que compõem a configuração base; *Planned* - circuitos planejados; etc.).

A Figura 6.6, por sua vez, descreve os atributos de *Network*. Tais atributos conceituam os elementos relevantes na representação de uma rede de transmissão. O nível de detalhamento dependerá do escopo do problema. No presente caso, foram definidas as barras terminais de um circuito da rede de transmissão, o limite de carregamento e o carregamento em termos de

potência ativa e o número de circuitos que um ramo deste nó permite introduzir em paralelo (faixa de passagem).

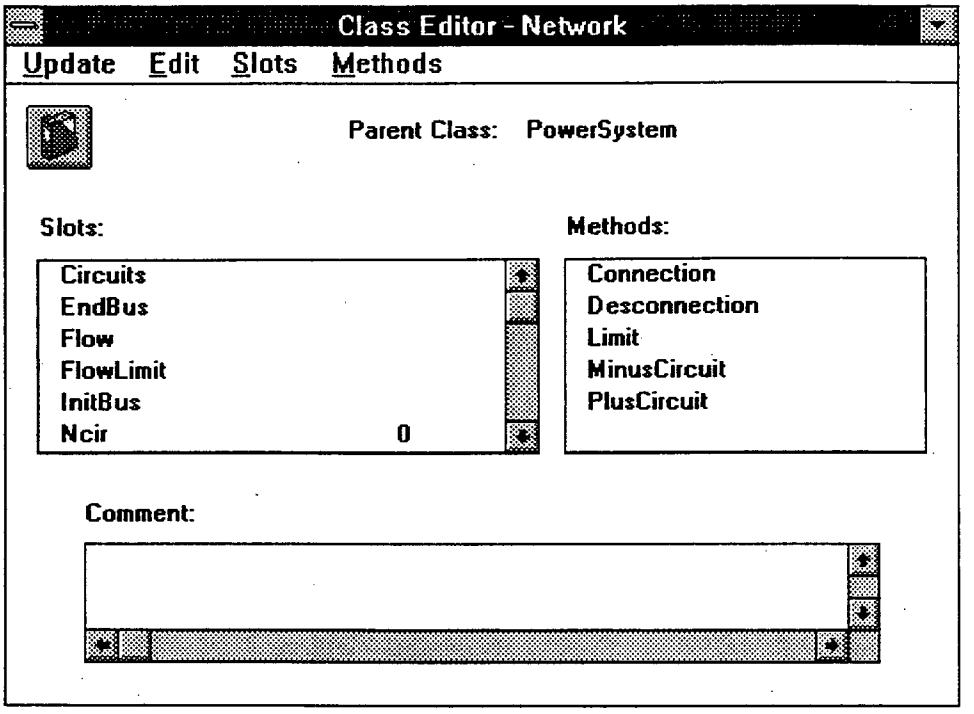


FIGURA 6.6 - ATRIBUTOS DE UMA DADA CLASSE

A Figura 6.7 apresenta o corpo de um método da classe *Network*, o método *Limit*, responsável pelo monitoramento do carregamento de circuitos em termos de potência ativa. Este método se prende ao slot *Flow*. Caso haja sobrecarga, isto é, o novo valor do slot *Flow* supera o valor do slot *FlowLimit*, o *flag* associado muda de estado.

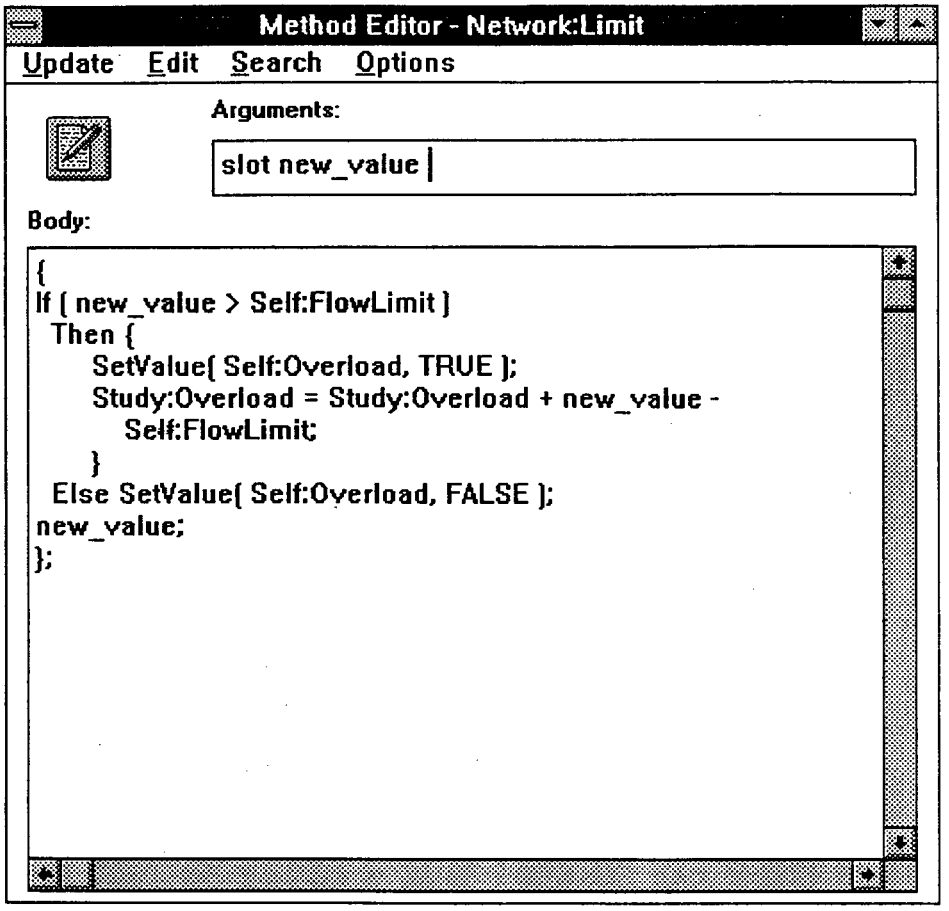


FIGURA 6.7 - MÉTODO ASSOCIADO A UM SLOT DA CLASSE NETWORK

Como segundo exemplo do uso de métodos, o slot *Ncir* diz respeito ao número atual de circuitos na faixa de passagem *Circuits*. O sistema não permite que se coloquem mais circuitos do que o permitido. O controle é feito por dois métodos, *MinusCircuit* e *PlusCircuit*, que removem e adicionam circuitos, respectivamente, alterando o valor do slot *Ncir*. Quando este chega ao limite (valor contido em *Circuits*), um aviso é dado ao usuário.

Esta foi a representação de conhecimentos adotada no protótipo SAPIENS. Estes e outros métodos, bem como os demais processos pertinentes ao modelo estão detalhados no Apêndice C.

6.5. CONCLUSÕES

As técnicas utilizadas para a implementação de sistemas baseados em conhecimento apresentam uma grande similaridade em relação aos métodos de otimização empregados pelos planejadores nos estudos de redes elétricas, e apresentam uma solução promissora para os problemas encontrados na atual metodologia do planejamento da expansão de sistemas de transmissão de energia elétrica, uma vez que possibilitam a representação das heurísticas de planejamento de uma forma estruturada, inteligível, e relativamente simples de serem implementadas, o que facilita seu acesso e atualização pelos usuários.

O uso das técnicas de inteligência artificial abrem novas perspectivas para a melhoria da interface usuário/programas de planejamento, o que se deve à possibilidade de se criar meios de diálogo em linguagem natural. Álem disso, a troca da base de conhecimentos fica facilitada quando se quer estudar sistemas de transmissão cujas características não correspondem às regras originais.

Com as ferramentas de software que se dispõem hoje em dia, ainda não é possível automatizar por completo o processo de planejamento. O mesmo apresenta grande complexidade em sua formulação (vide Capítulo2). Contudo, com o emprego adequado de técnicas de IA, é possível desenvolver pequenos módulos de sistemas baseados no conhecimento que melhorem o desempenho dos modelos atuais, já validados, trabalhando em conjunto com os mesmos e criando um ambiente híbrido para simulação de cenários de expansão.

CAPÍTULO 7- VERSÕES IMPLEMENTADAS

7.1. INTRODUÇÃO

Graças ao emprego da metodologia apresentada no Capítulo 6, foi implementado o protótipo *SAPIENS* (*Sistema Adaptativo e Interativo de Planejamento da Expansão*), um sistema baseado no conhecimento para apoio ao processo de planejamento de sistemas de transmissão.

A escolha do nome SAPIENS advém do caráter evolutivo que o modelo contém, desenvolvendo-se conforme as necessidades dos seus usuários. Esta característica é inerente a sistemas baseados no conhecimento, devido à modularidade entre os seus elementos constituintes: métodos, funções e base de conhecimentos (fatos, regras de produção e objetos - vide Capítulo 4).

O objetivo principal deste capítulo é, pois, ressaltar a flexibilidade do protótipo SAPIENS frente aos seus predecessores, descrevendo suas primeiras etapas de implantação e o seu aprimoramento.

A fim de elucidar o caráter evolutivo do protótipo SAPIENS, serão apresentadas, de forma sucinta as diferentes versões implementadas.

7.2. SAPIENS 1.0 - PROTOPROTÓTIPO

A primeira versão do modelo SAPIENS foi uma *cópia* do modelo SINTRA, da Eletrobrás. Nada mais se fez do que montar uma versão que pudesse operar no ambiente Windows Microsoft Inc, sem haver necessidade de drásticas mudanças na arquitetura do software fonte original.

Em sua primeira versão (Figura 7.1), o protótipo SAPIENS não possuia cérebro, sendo visto mais como um animal adestrado, instintivo, que executava todas as tarefas expedidas pelo seu dono. O SAPIENS apenas passava comandos ao SINTRA (modelo básico) através de um arquivo de comandos gerado conforme as necessidades dos usuários.

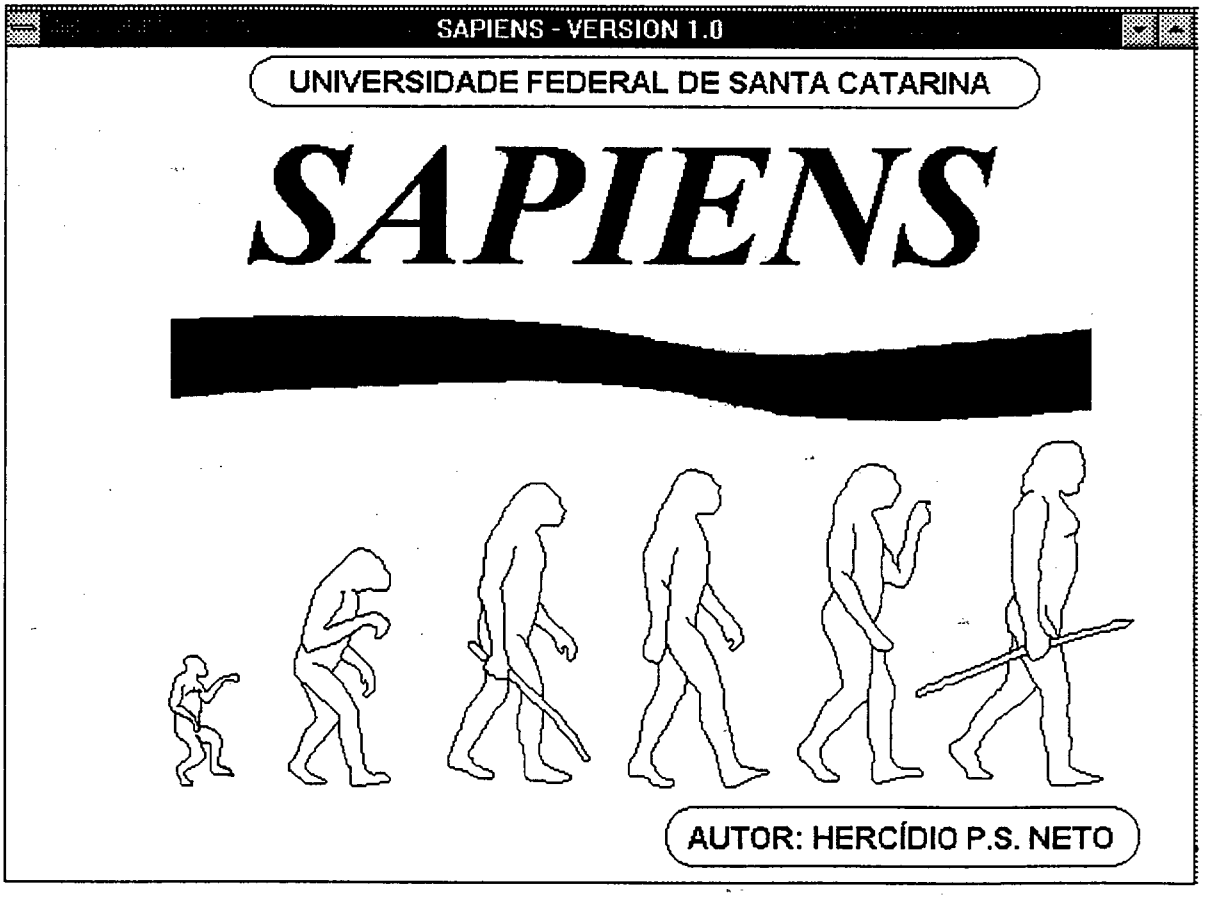


Figura 7.1 - SAPIENS 1.0 - janela de abertura

Todos os comandos do modelo SINTRA (síntese, análise e gerais) são executados sob uma interface composta de menus e janelas (Figura 7.3). Estas só se tornavam ativas após terem sido

definidos os arquivos de dados do caso base e de adições (Figura 7.2). Foram providenciadas janelas (*Transcript Windows*) para visualizar o conteúdo dos arquivo de dados, o arquivo de resultados e o arquivo de comandos, os dois últimos sempre em expansão. A simulação do comportamento estático da rede em estudo ficava a cargo do modelo SINTRA modificado, que era executado em *shell* com o DOS.

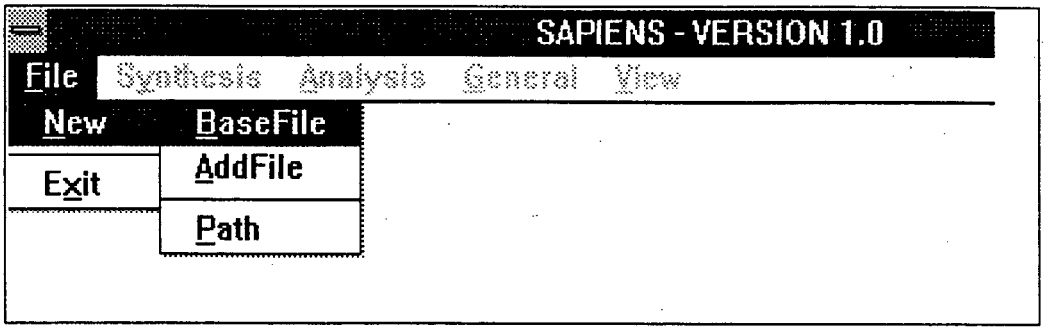


Figura 7.2 - Definição dos arquivos de dados

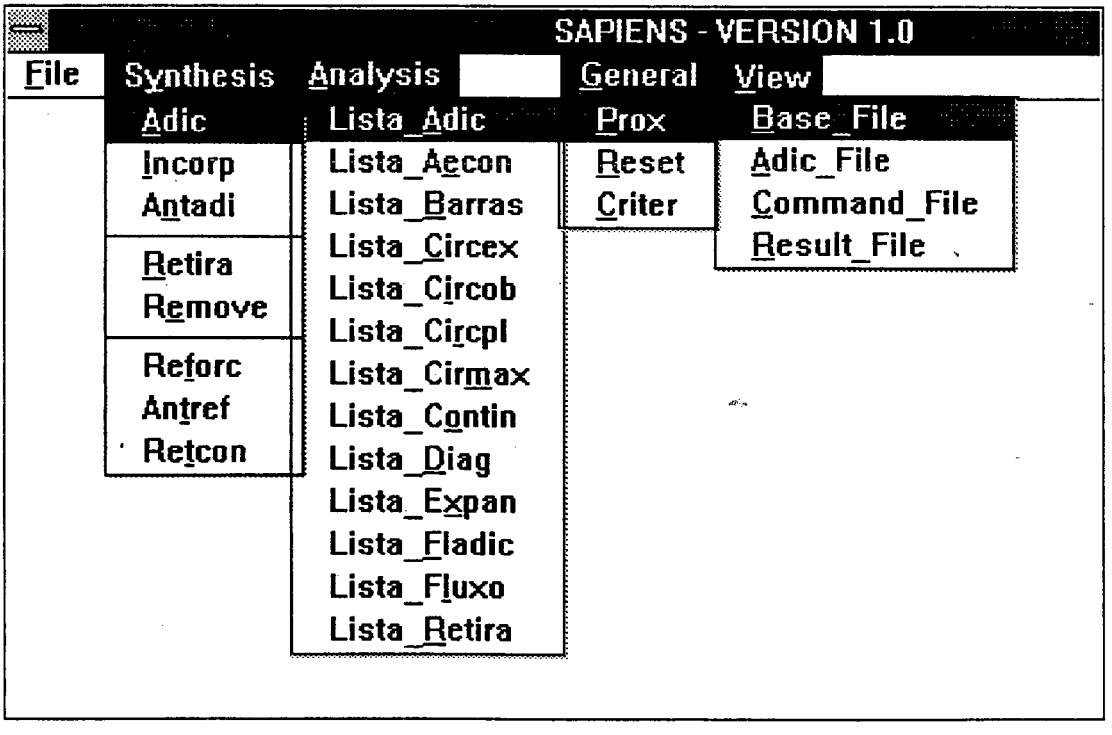


Figura 7.3 - Janelas e menus de comandos

O modelo SINTRA foi modificado para possibilitar a leitura de seus comandos a partir de um arquivo e não do teclado. Este modelo passou pois a interagir com o SAPIENS, criando-se três níveis para o estudo de cenários de expansão: PLANEJADOR - SAPIENS - SINTRA.

A Figura 7.4 apresenta o fluxograma simplificado de como opera a versão 1.0 do protótipo SAPIENS. Vale frisar que com esta versão é possível registrar o cenário de expansão obtido para um dado estudo. Este cenário encontra-se armazenado no arquivo de comandos gerado com o auxílio do protótipo SAPIENS. O acompanhamento da expansão se faz através do próprio arquivo de resultados, que registra todas ocorrências.

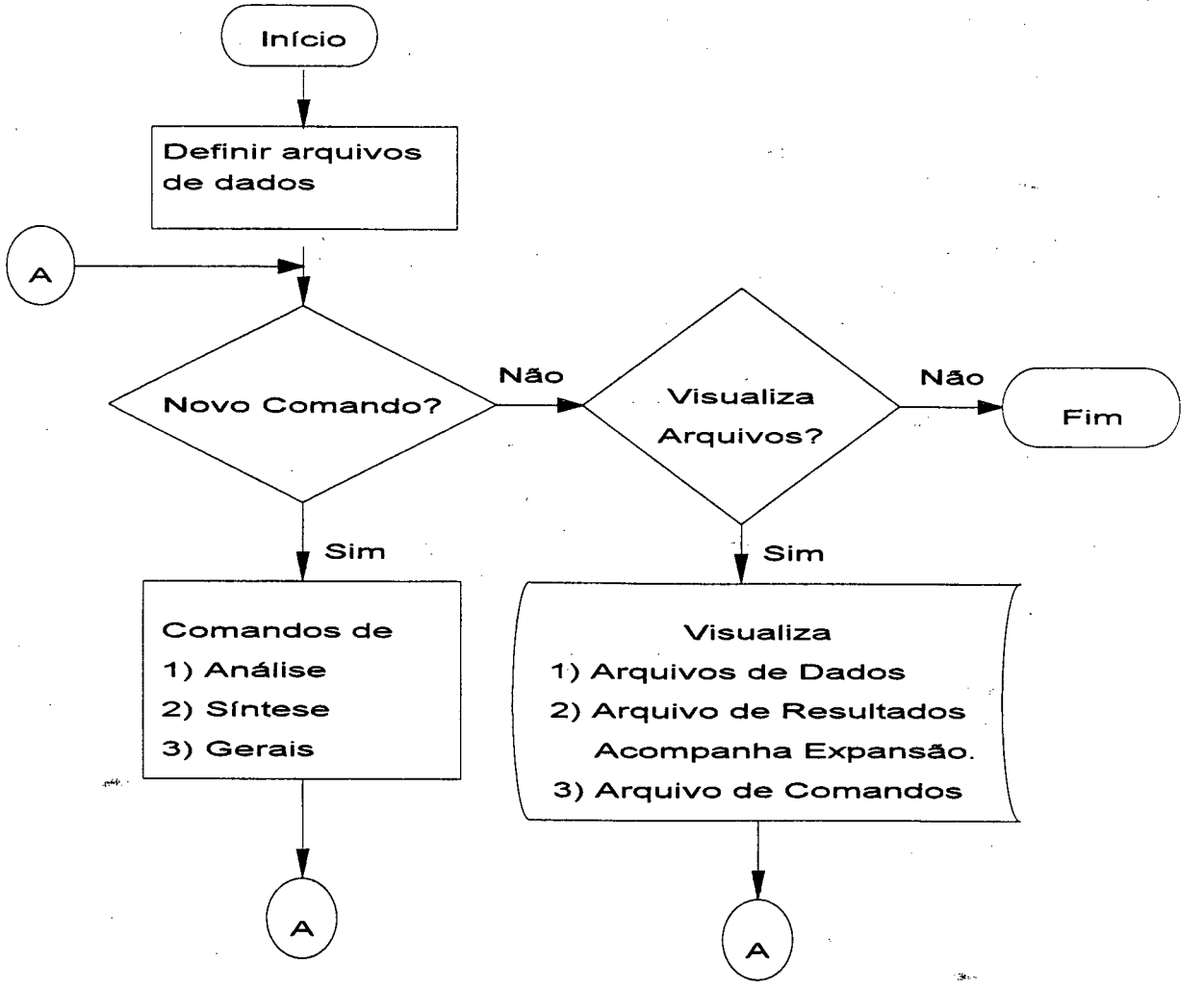


Figura 7.4 - Fluxograma da versão 1.0

7.3. VERSÃO 1.1 - VERSÃO PARA DIAGNÓSTICO

Em sua segunda versão, o modelo SAPIENS começou a se tornar mais *inteligente*, realizando diagnósticos para uma dada rede de transmissão, informando os *sintomas clínicos* a que o modelo se atém são, conforme Figura 7.5:

1. balanço de potência ativa;

2. conectividade entre os nós com injeção líquida de potência ativa diferente de zero;

3. sobrecarga nos ramos de transmissão (LT's e trafos).

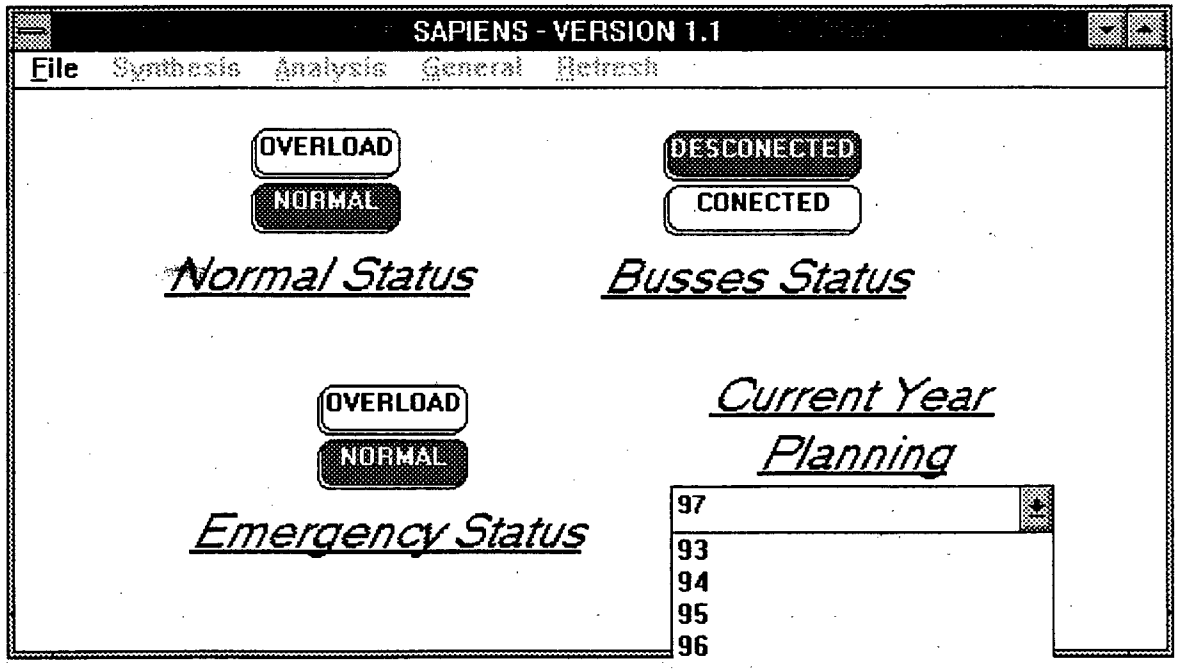


Figura 7.5 - Versão para diagnóstico de redes de transmissão

Tais análises, ou diagnósticos, tendem a expandir conforme o nível de detalhamento desejado para estudo (vide Capítulo 2). A base de conhecimentos é modelada para atender a cada um desses tipos de análise citados.

Nesta segunda versão, fez-se uso de duas *DLL*'s (Dynamic Link Libraries)*. Uma DLL é dedicada à leitura dos dados dos arquivos do caso base (configuração inicial da rede de transmissão) e dos dados do arquivo de adições (circuitos planejados para compor o programa de expansão), montando assim os fatos da base de fatos. Uma segunda DLL atualiza a base de fatos com a leitura dos dados de mercado (geração/carga) para o ano de planejamento corrente.

A Figura 7.6 apresenta o fluxograma simplificado de operação da versão 1.0 do protótipo SAPIENS. Como diferenças básicas, cita-se o módulo de diagnóstico da rede em estudo, bem como o comando de definição de novo ano de estudo, que deixou de ser ativado via menu para se tornar em um comando via mouse. Enquanto a primeira inovação é importante do ponto de vista de inclusão de novas heurísticas de planejamento a modelos existentes, a segunda é importante do ponto de vista ergonômico, uma vez que é mais fácil para o operador trabalhar com dispositivos apontadores do que com o teclado. Em sistemas gráficos-multitarefas, o uso do mouse facilita o trabalho com diferentes pacotes concomitantes. Por exemplo, uma seção de trabalho pode levar o usuário a ter em aberto um editor de texto, uma planilha de cálculo e um editor gráfico ao mesmo tempo, sendo o mouse um dispositivo adequado para se intercambiar entre estes diferentes ambientes de trabalho.

Nestas duas versões, o motor de simulação do protótipo, o sistema SINTRA, era executado a cada novo comando que lhe era passado.

---

* Rotinas pré-compiladas responsáveis pela montagem e atualização da base de fatos (as instâncias que definem a rede de transmissão em estudo) do protótipo durante uma seção de estudo. As DLL's diferem das bibliotecas de funções pré-definidas pelo fato de poderem ser utilizadas como subrotinas no corpo de qualquer outro programa, independente da linguagem utilizada pelo compilador deste (i.e.).

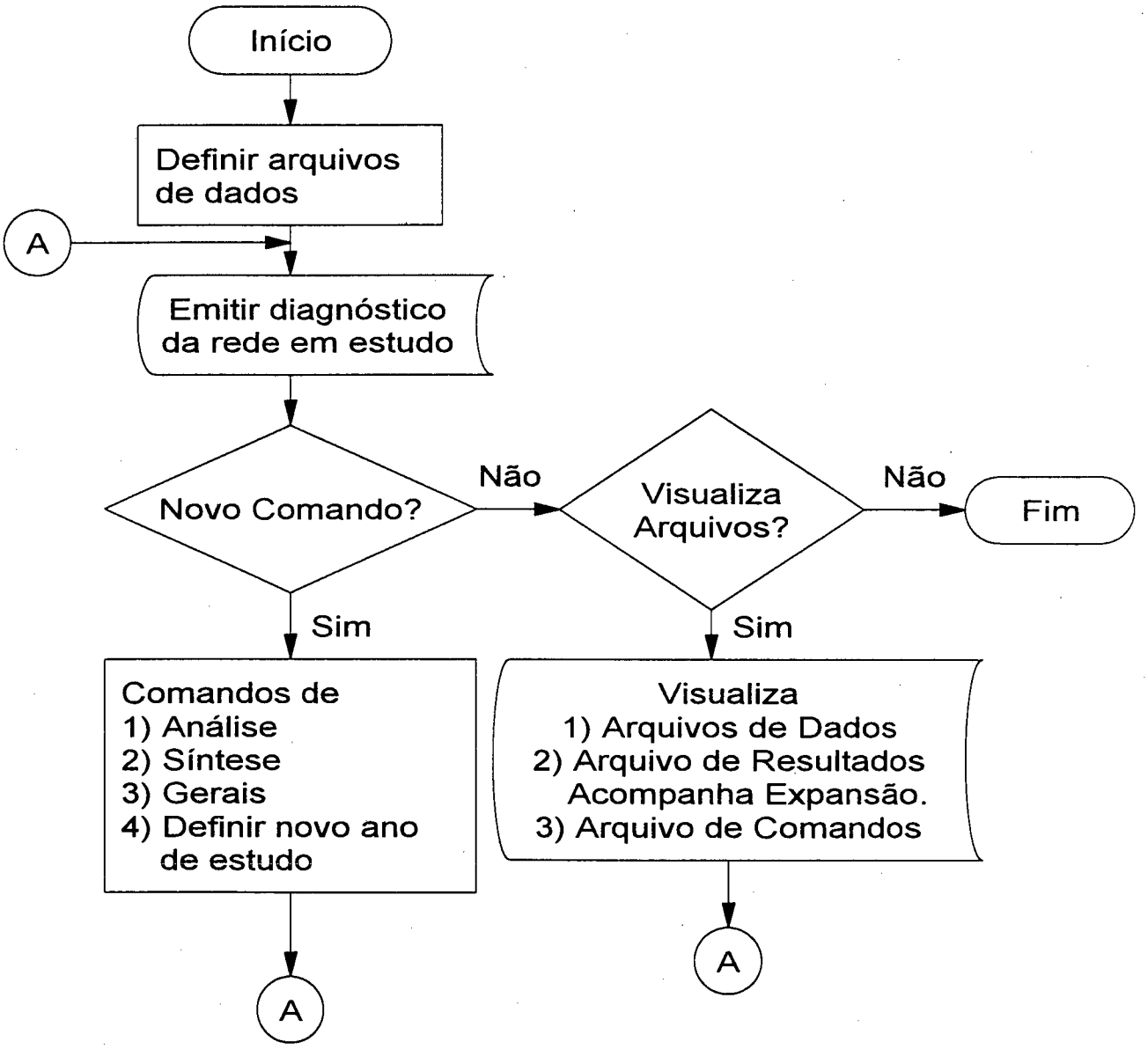


Figura 7.6 - Fluxograma da versão 1.1

Até aqui, as duas versões apareceram em inglês. Para tornar o protótipo mais conversacional, optou-se pelo português. Mesmo a tradução foi uma tarefa simples, uma vez que o *shell* KAPPA-PC, gera um código fonte em linguagem C, representativo do protótipo modelado. Bastou pois trabalhar neste arquivo com um editor de texto padrão ASCII.

Por estar subordinado ao *shell* KAPPA-PC, os protótipos atuais são *interpretados* pelo mesmo. Outra vantagem de se ter o código fonte em linguagem C, é que este pode ser compilado, gerando um programa executável. Este tornaria os protótipos portáveis, pois não dependeriam da

presença do *shell* KAPPA-PC para serem operados. Para efetivar este intento é preciso ter uma versão comercial do KAPPA-PC, Intellicorp Inc.

7.4. SAPIENS 2.0 - VERSÃO ATUAL

A terceira versão é a mais atual. Como seus antecessores, também usa rotinas do SINTRA como seu motor de simulação. Busca incorporar os elementos modulares de forma independente, realizando o paradigma OOP.

A base de conhecimentos é semelhante nas três versões, pois durante a sua implementação, buscou-se levar por igual os melhoramentos atingidos. Contudo, a terceira versão possui em sua forma mais lógica, separando os elementos constituintes de um sistema de transmissão em classes de objetos adequados.

7.4.1. Ambiente de Trabalho

Conforme segue pela Figura 7.7 uma janela inicial mostra o título do protótipo. Esta contém dois *Buttons*, um para iniciar uma nova seção de trabalho e outro para finalizar os estudos.

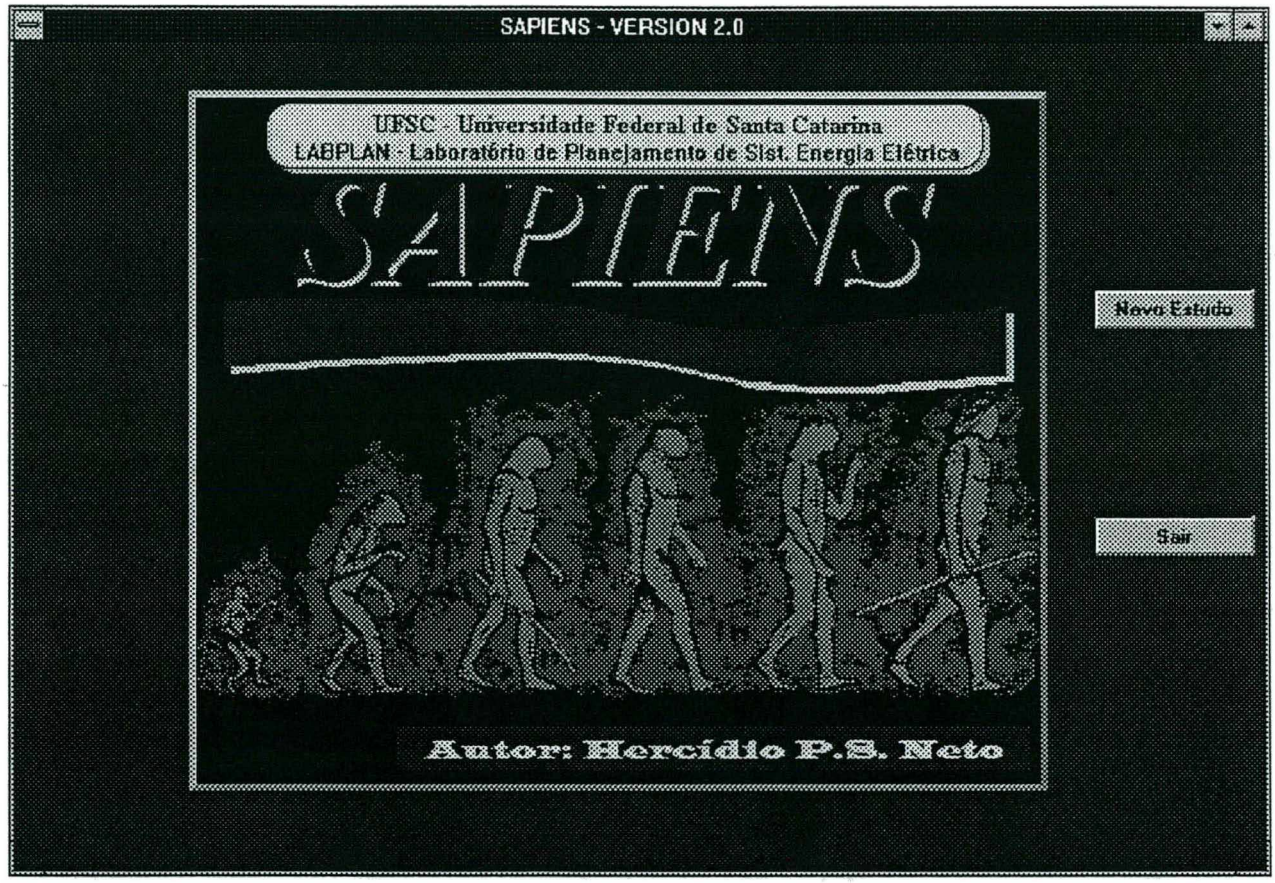


Figura 7.7 - Janela de abertura

7.4.2. Seção de estudo

a) Arquivos de dados

Ao iniciar *Novo Estudo*, o SAPIENS exige que se definam quais são e onde estão os arquivos de dados necessários à configuração do sistema de transmissão (Figura 7.8), quais sejam: arquivo de caso base, arquivo de adições e arquivo de coordenadas de tela; este último define a topologia da rede para visualização no vídeo (Figura 7.9).

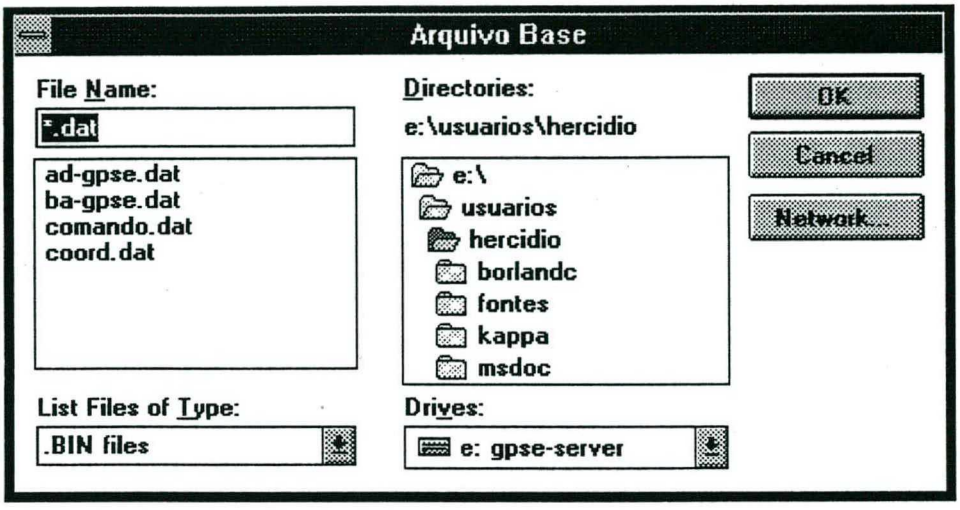


Figura 7.8 - Definição dos arquivos de dados

Estes três arquivos definem o caso de estudo e habilitam as outras funções do protótipo. No video, é apresentado um diagrama unifilar contendo a topologia da rede de transmissão. Os circuitos que compõem a configuração base têm linha cheia e cor preta. Os circuitos planejados têm linha pontilhada e cor lilás.

Uma desvantagem do uso do shell KAPPA-PC, é sua limitação de matizes de cores. Até a presente versão, ele trata apenas com oito matizes: branco, preto, azul, verde, amarelo, lilás, vermelho e marrom (ver [29]).

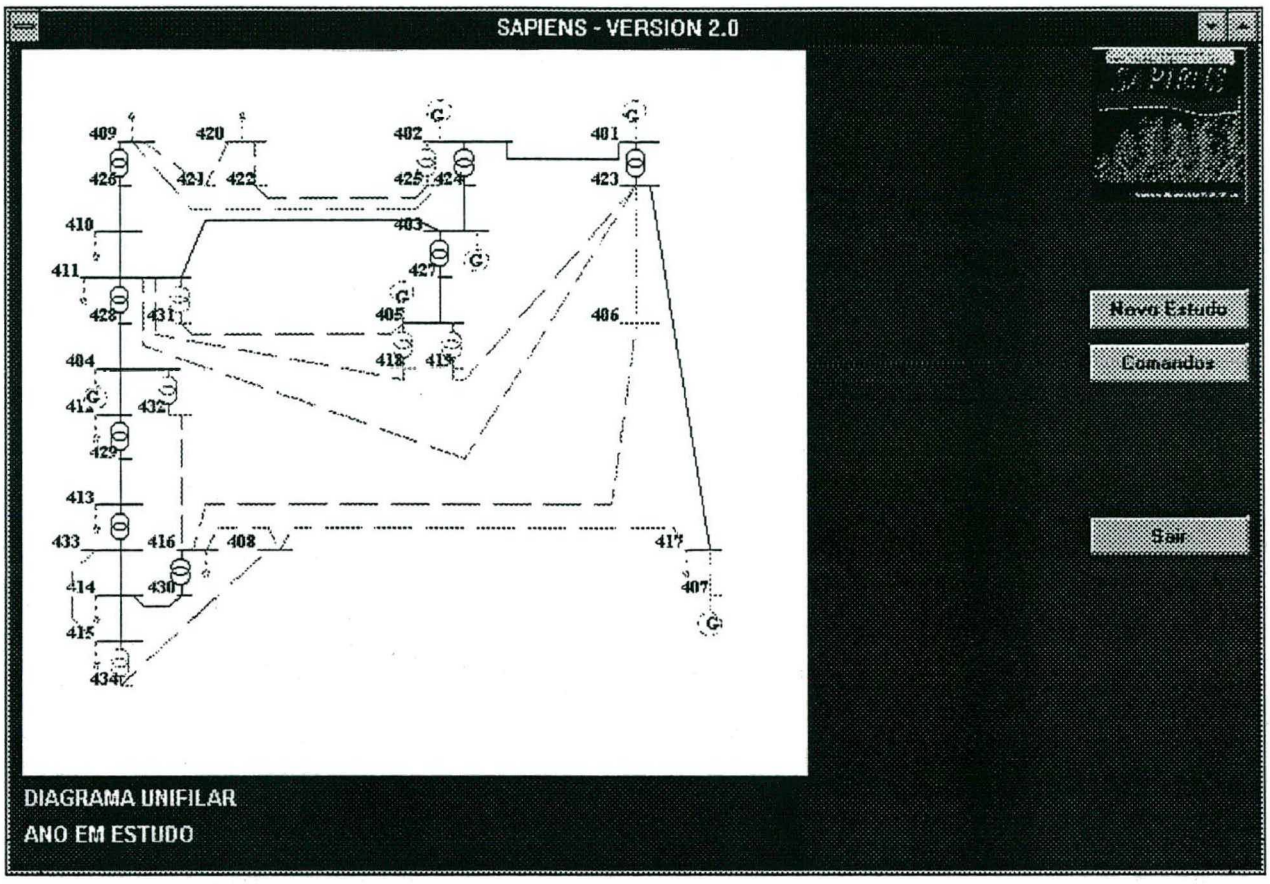


Figura 7.9 - Estado inicial de estudo

Surge um terceiro *Button*, *Comandos* (Figura 7.9), responsável pelo processo de planejamento em si. Através deste *Button* o usuário tem acesso aos comandos de análise e síntese que o SINTRA provê.

O primeiro comando a ser dado é justamente a definição do ano de planejamento dentro do horizonte de estudo definido (Figura 7.10).

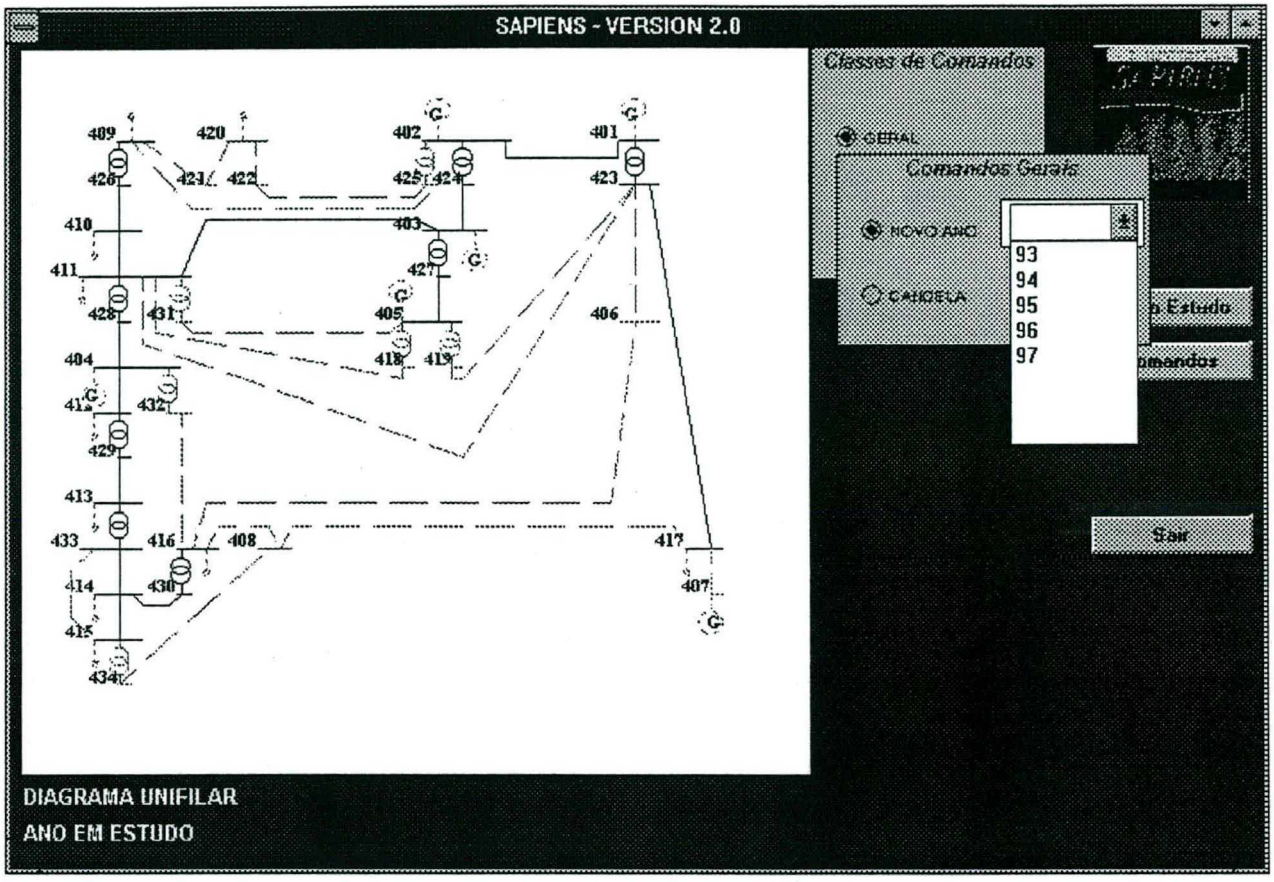


Figura 7.10 - O primeiro comando

Após ter determinado o ano de estudo, o SAPIENS emite um diagnóstico sucinto sobre o estado do sistema quanto à presença de barras desconectadas com injeção líquida de potência diferente de zero (Figura 7.11). Este é apenas um alerta para que o usuário dê preferência aos circuitos que incorporem estas barras desconectadas ao sistema de transmissão. O usuário é livre para agir, podendo ou não integrar os novos perfis de mercado (geração/carga) ao sistema em estudo.

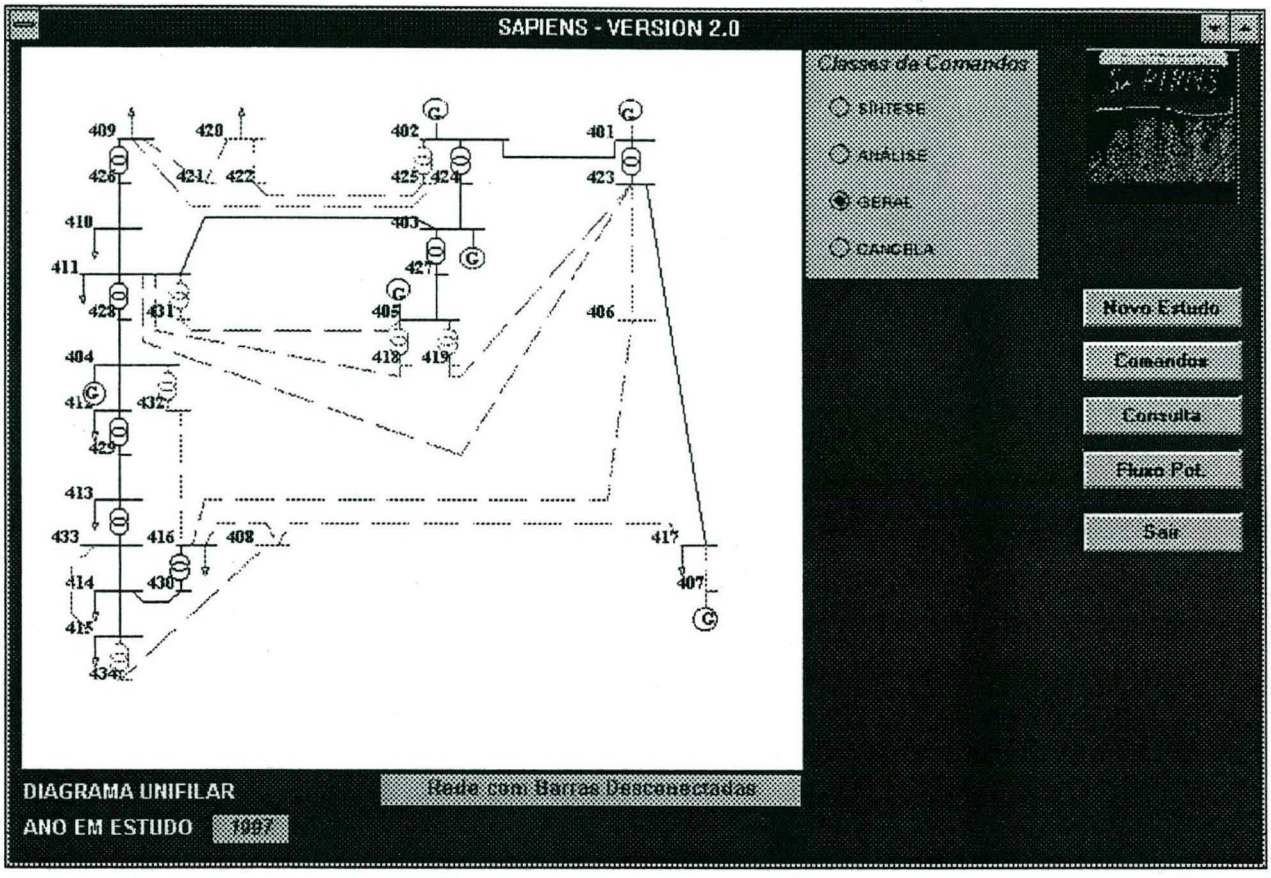


Figura 7.11 - Diagnóstico da rede

b) Comandos de Simulação

Somente após a definição do primeiro ano de estudo é que ficam habilitados os demais comandos do protótipo.

O *Button Comandos* abre um *RadioGroupButton* contendo como opções, três classes básicas de comandos empregadas pelo SINTRA: comandos de síntese, comandos de análise e comandos gerais.

i. Comandos de Síntese

Pela prática adquirida em estudos de planejamento, determinou-se que os comandos de síntese incluiriam apenas a síntese manual e automática, realizando tarefas de adição e remoção de circuitos. Não se incluiu no protótipo comandos de síntese para contingências severas, nem se utilizou outros tipos de circuitos que não fossem linhas de transmissão e transformadores. Com

isto, buscou-se reduzir o escopo de aplicação do protótipo, como se faz para todo sistema baseado no conhecimento.

Cada elemento presente no diagrama unifilar da rede em estudo é um objeto.

Ao selecionar adição ou retirada manual, é apresentado um *ComboBox* contendo todos os possíveis circuitos adicionáveis, mesmo os que já foram adicionados, pois pode haver duplicação de circuitos (Figura 7.12). Ao atingir o limite da faixa de passagem de um circuito, ou quando não houver circuitos a retirar, mensagens informam o usuário para selecionar outro circuito* (Figura 7.13). Ao selecionar um circuito para adição, o mesmo muda de cor no diagrama unifilar, sugerindo ao usuário que confirme sua escolha antes de incluí-lo ou não no presente plano de expansão. Ao selecionar um circuito para exclusão, o mesmo é apagado do vídeo, aguardando do usuário confirmação para retirar o referido circuito do plano de expansão.

Na síntese automática, tanto a adição, quanto a retirada de circuitos segue de forma transparente ao usuário. Este apenas indica a quantidade de circuitos a adicionar ou retirar automaticamente, por meio de um *Meter* (Figuras 7.14 e 7.16). O acompanhamento se faz pela consulta ao arquivo de resultados. Este é mostrado no video na forma de arquivo texto, dentro de uma janela *Transcript* (Figuras 7.15 e 7.17).

---

* Este método que monitora os limites das faixas de passagens dos circuitos da rede em estudo, não existe no modelo original, o SINTRA, sendo um exemplo de que se pode incluir novas heurísticas a modelos de planejamento sem haver necessidade de se mudar o software fonte original.

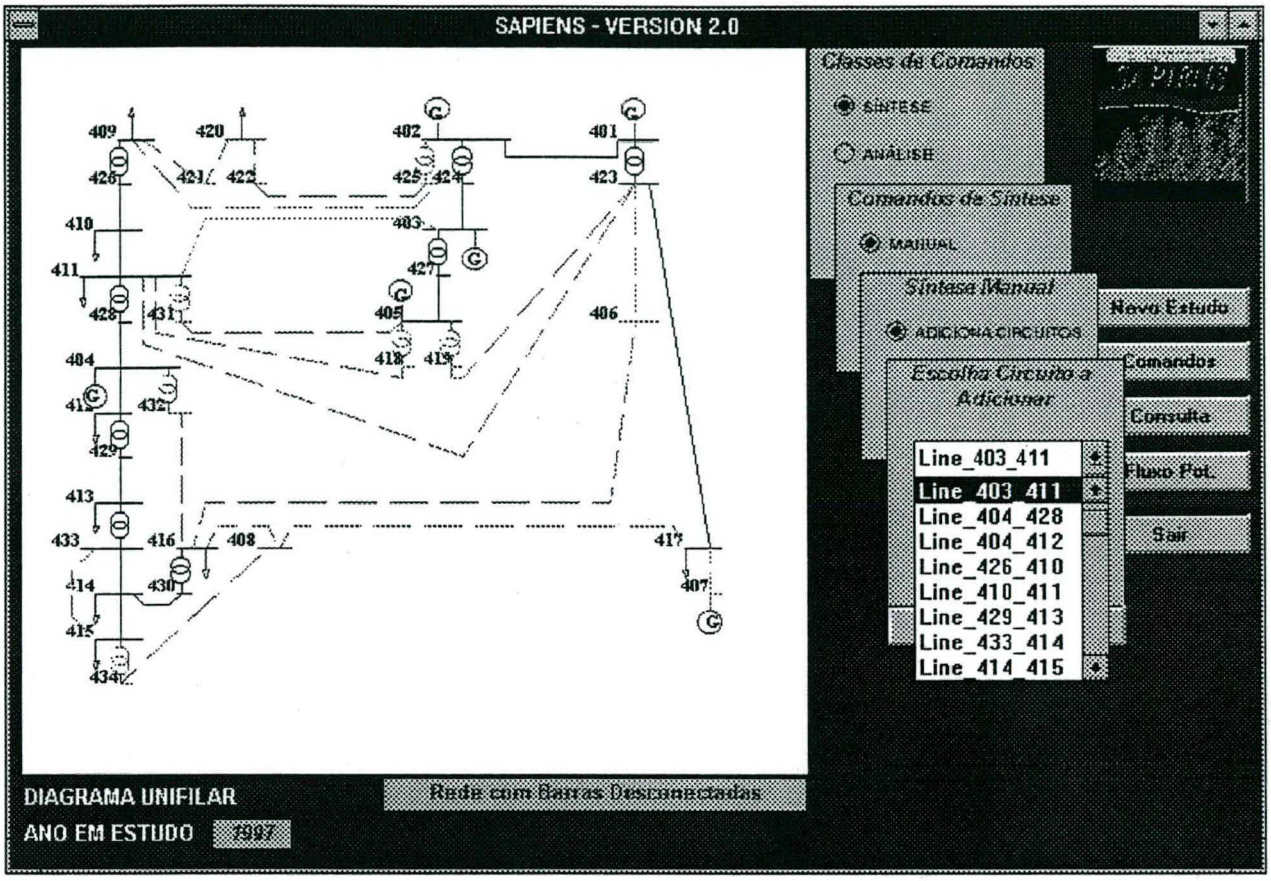


Figura 7.12 - Adição manual

Figura 7. - Mensagens de diagnóstico

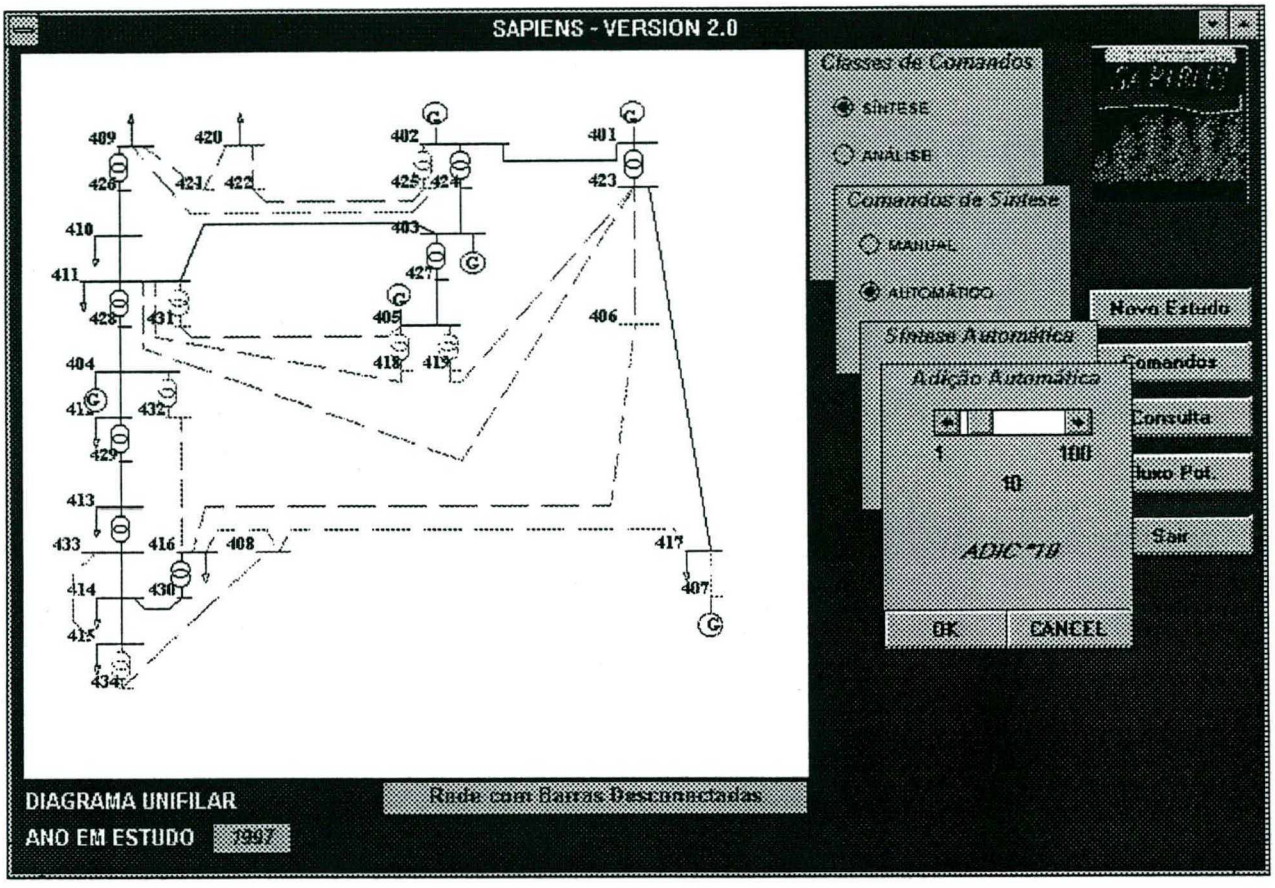


Figura 7.14 - Adição automática

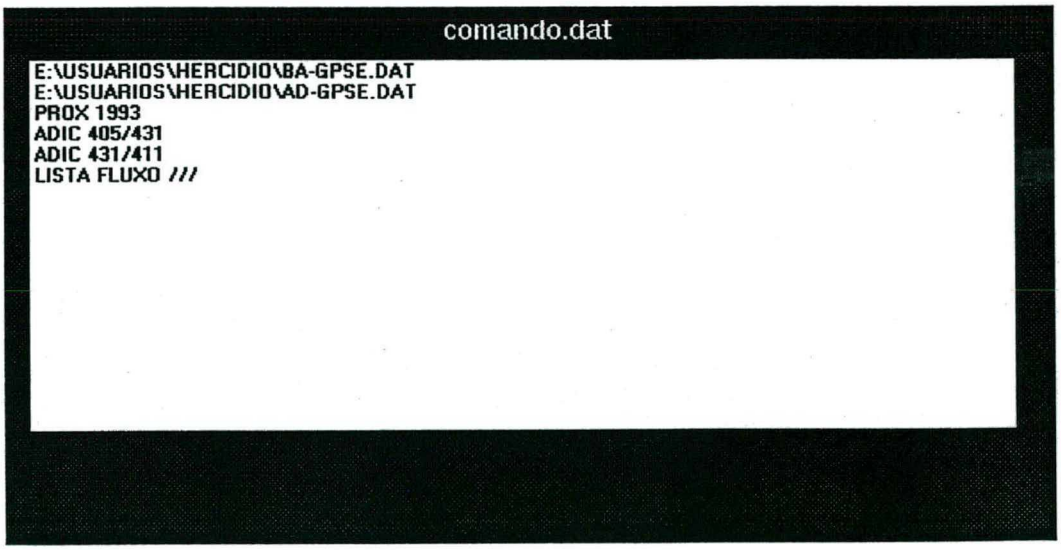


Figura 7.15 - Transcript mostrando conteúdo do arquivo de comandos

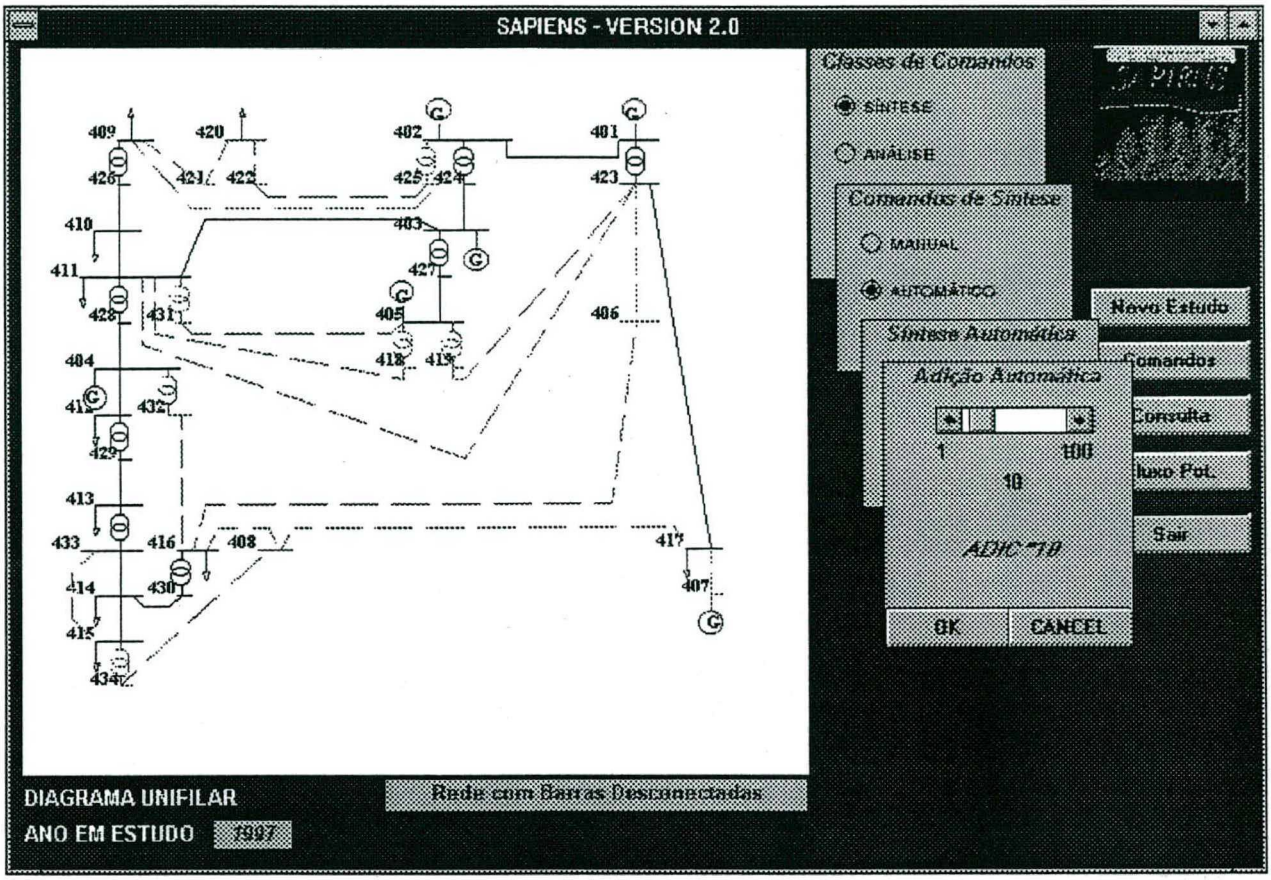


Figura 7.16 - Retirada automática

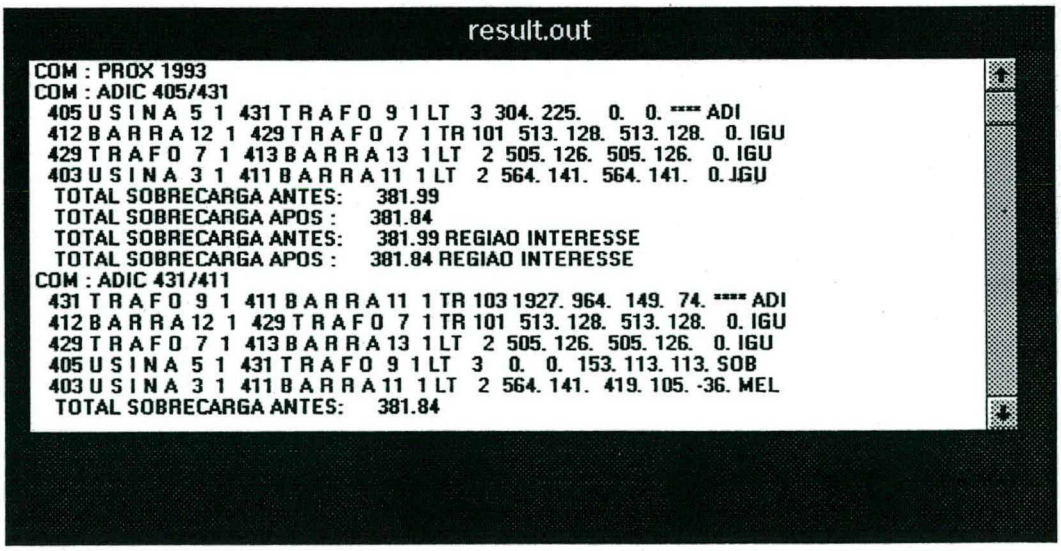


Figura 7.17 - Transcript mostrando conteúdo do arquivo de resultados

ii. Comandos de Análise

A opção para comandos de análise ativa um *RadioGroupButton* cujas opções são justamente os comandos de análise do modelo SINTRA, escritos de uma forma mais coloquial (Figura 7.18). Até o momento, esta foi a forma mais direta de inclui-los no protótipo.

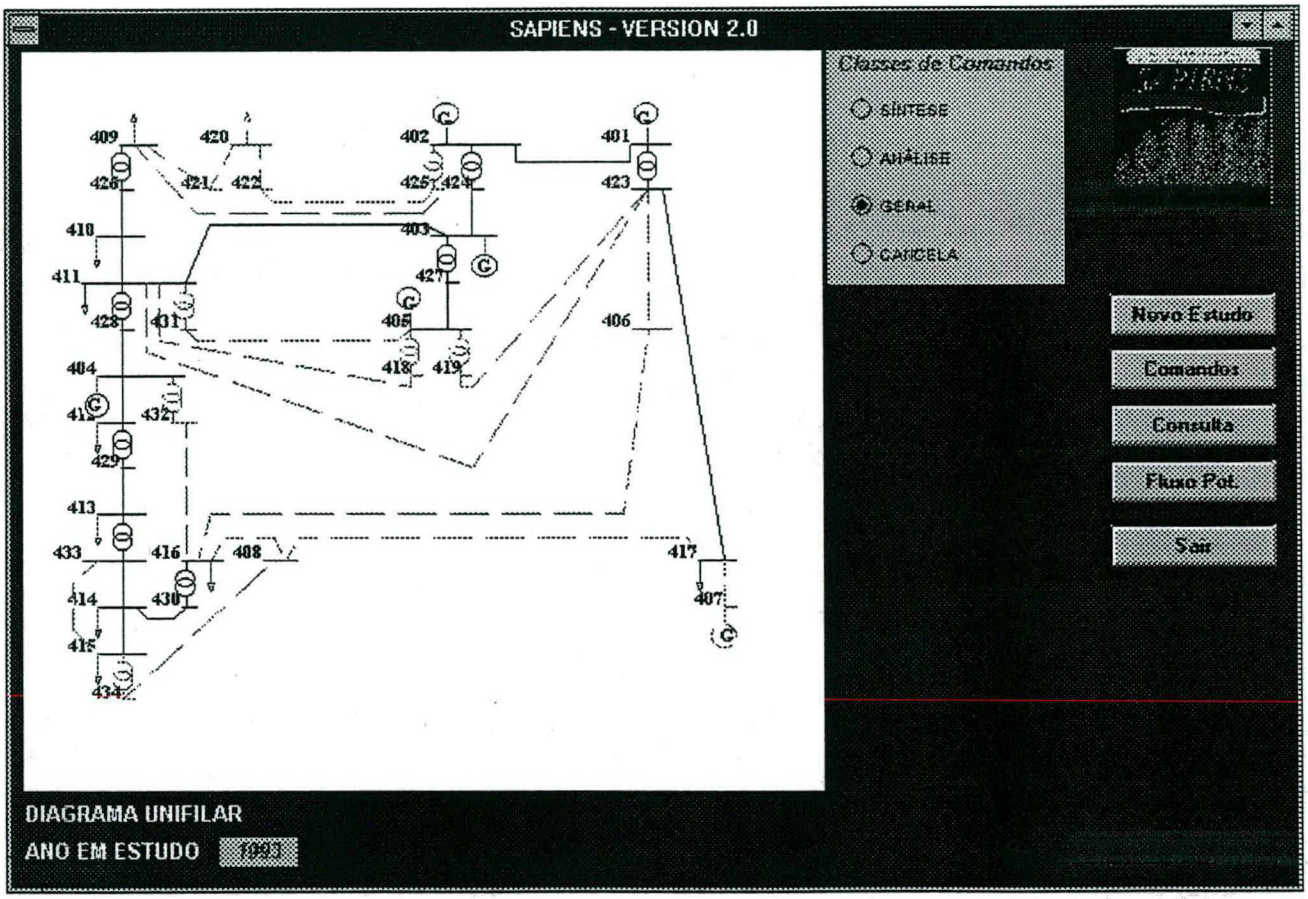


Figura 7.18 - Comandos de análise

O propósito do protótipo SAPIENS é se tornar, em última instância, um sistema adaptativo, flexível e modularizado, o que facilita sua manutenção e o desenvolvimento de versões mais aprimoradas. A opção por comandos via *mouse* na versão final advém do fato de que esta é a melhor solução do ponto de vista ergonômico.

iii. Comandos Gerais

Abre um outro *RadioGroupButton* contendo comandos gerais como no programa SINTRA. Entretanto, para simplificação de projeto, o comando CRITER (alteração de critérios para síntese automática) não está disponível até o momento.

Tem como opções escolher novo estágio de estudo; reinicializar a seção de estudo e simular os comandos selecionados. Todas opções são acessadas por *mouse*.

Ao selecionar um novo ano de estudo, é aberta uma janela *ComboBox* contendo os possíveis anos de estudo. Ao selecionar um dos mesmos, uma DLL se encarrega de atualizar a base de conhecimentos com os dados de mercado (geração/carga) para aquele período. Graficamente, os pontos (barramentos) com injeção de potência diferente de zero, apresentam-se com linha cheia e cor azul. O usuário acompanha visualmente esta alteração podendo situar-se geograficamente com os dados de estudo. O comando *Reinicializar* restitui a rede para seu estado inicial, cancelando todos os comandos previamente definidos.

Nenhum destes dois comandos (novo ano de estudo; reinicializar) ativa o motor de simulação. Este é ativado manualmente, via *mouse*, a fim de dar mais liberdade ao usuário*.

Para saber o resultado da simulação, é preciso acionar o *Button* Fluxo de Potência. Este faz uso de uma DLL que lê o arquivo de resultados em busca dos novos valores de fluxo de potência. A mesma DLL atualiza o diagrama unifilar apresentando os circuitos em sobrecarga de potência ativa (Figura 7.19).

Este comando, *Fluxo de Potência*, na verdade inclui o comando de análise <LISTA FLUXO ///> ao fim da lista de comandos de simulação. Monta-se o arquivo de comandos com esta lista, abre-se um shell para o DOS e é executado o programa SINTRA. Uma DLL lê o

---

* A versão atual tem como maior limitação o fato de ter como motor de simulação um programa *for DOS*. Para executá-lo, o Windows abre um *shell* para DOS, priorizando esta atividade em detrimento de todas as outras que estiverem sendo executadas. Para minimizar o tempo de espera do usuário enquanto o sistema simula o plano de expansão proposto, optou-se por deixar por conta do usuário a ativação do motor de simulação.

arquivo de resultados e métodos verificam se algum circuito existente, ou adicionado está com sobrecarga de potência ativa, isto é, se o seu carregamento está acima do nominal.

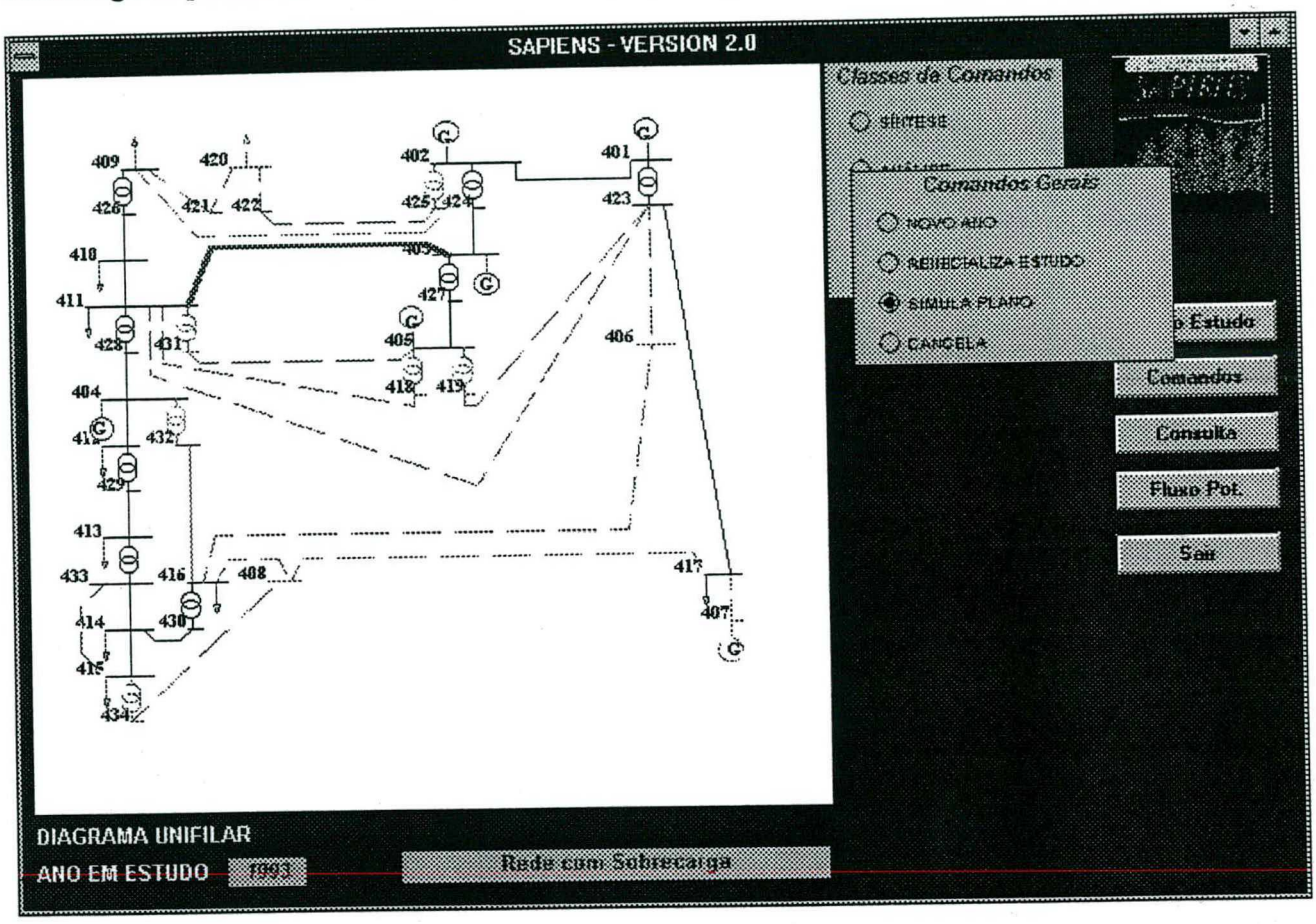


Figura 7.19 - Análise de fluxo de potência

c. Sair

Finaliza a seção de trabalho e reexibe a janela de apresentação.

A Figura 7.20 apresenta o fluxograma simplificado da Versão 2.0. Esta versão está mais estruturada do que as precedentes. Nesta versão, há uma nítida separação das classes que definem fisicamente o sistema de transmissão e os comandos de simulação para o planejamento da expansão. O sistema está mais modular.

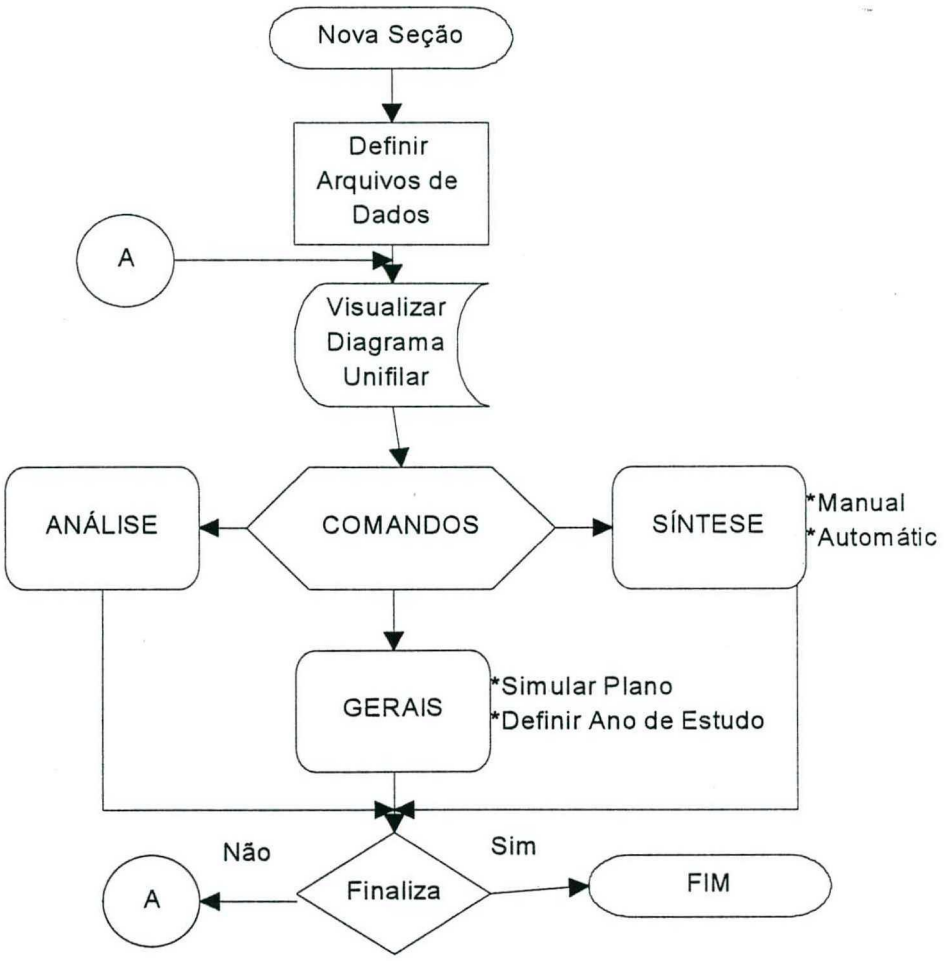


Figura 7.20 - Fluxograma da versão 2.0

7.5. CONCLUSÕES

As três versões aqui dispostas como frutos deste trabalho, estão longe de compor versões finais. Servem apenas como protótipos, e atendem a um escopo limitado de análise, como ficou claro no capítulo anterior.O próximo capítulo mostra um exemplo de aplicação do protótipo em um estudo de planejamento da expansão. O sistema teste escolhido corresponde à configuração do sistema da Região Sul simplificado.

CAPÍTULO 8 - ESTUDOS DE CASOS

8.1. INTRODUÇÃO

O presente capítulo apresenta um estudo de caso, com o objetivo de avaliar o desempenho do protótipo SAPIENS, como modelo de apoio no planejamento de sistemas de transmissão.

8.2. CASO TESTE

Como caso teste, foi desenvolvido um caso hipotético com base no sistema de transmissão reduzido da região sul brasileira, modelado por [44], modificado por [42] e alterado novamente pelo presente trabalho pela inclusão de transformadores entre barramentos com níveis de tensão diferentes. Os dados referentes a estes transformadores foram colhidos de catálogos de concessionárias de energia elétrica.

As dimensões do caso teste estão listadas na Figura 8.1, gerada pelo programa SINTRA. Trata-se de um sistema de pequenas dimensões, mas que procura retratar algumas das configurações possíveis de ocorrer em um sistema de transmissão real. A seguir, a descrição da base de dados.

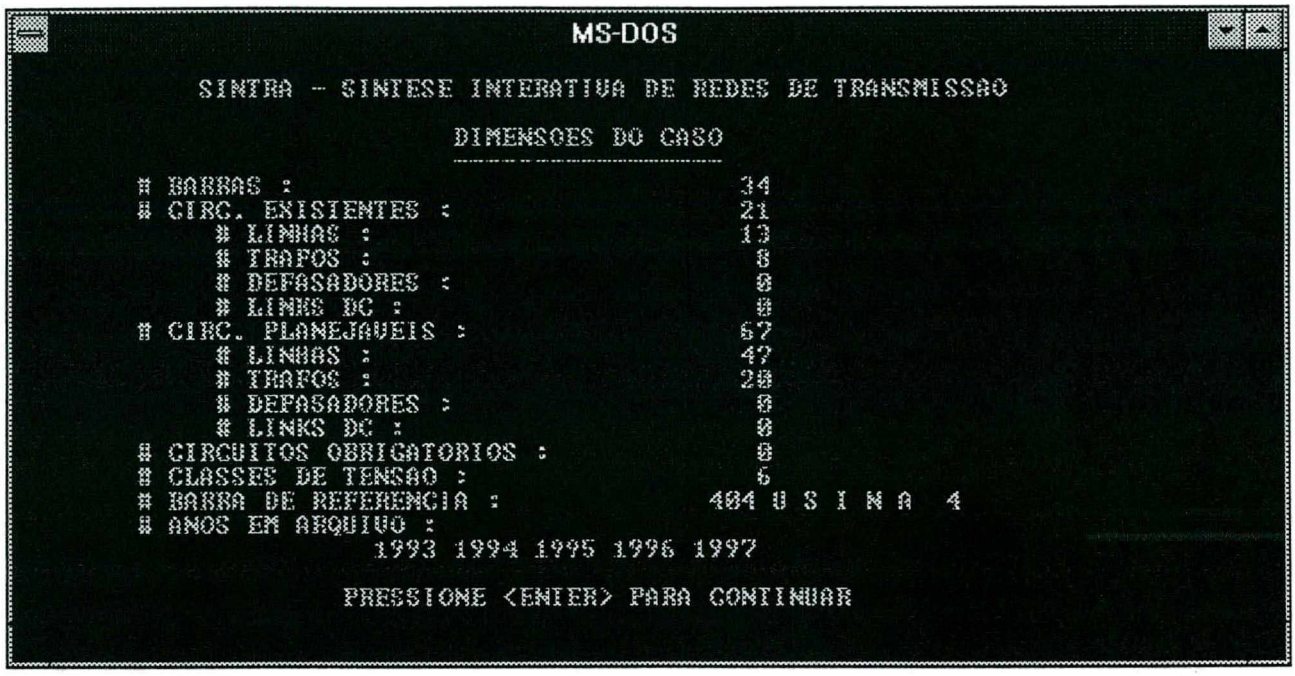


FIGURA 8.1 -DIMENSÕES DO CASO TESTE

8.2.1. Arquivo do caso base

O arquivo do caso base, ***ba-gpse.dat***, contém dados a respeito da configuração da rede em estudo para o ano inicial de planejamento. Como o protótipo SAPIENS, sua última versão, foi modelado para tratar apenas com linhas de transmissão e transformadores, não foram considerados outros tipos de circuitos neste exemplo* .

O Quadro 8.1 contém os dados de interesse ao protótipo SAPIENS, referentes aos barramentos da rede de transmissão, presentes no arquivo do caso base.

* ~~Transformadores defasadores, links~~ DC, etc.

QUADRO 8.1 - DADOS DE BARRA DO CASO TESTE

NÚMERO DA BARRA	NOME	AREA	TIPO (PQ, PV, Vθ)	NÍVEL DE TENSÃO (KV)
401	USINA 1	1	1	500
402	USINA 2	1	1	500
403	USINA 3	1	1	230
404	USINA 4	1	2	500
405	USINA 5	1	1	138
406	USINA 6	1	1	230
407	USINA 7	1	1	230
408	USINA 8	1	1	230
409	BARRA 9	1	0	138
410	BARRA 10	1	0	230
411	BARRA 11	1	0	230
412	BARRA 12	1	0	500
413	BARRA 13	1	0	230
414	BARRA 14	1	0	500
415	BARRA 15	1	0	500
416	BARRA 16	1	0	230
417	BARRA 17	1	0	230
418	FICTICIA A	1	0	230
419	FICTICIA B	1	0	230
420	BARRA 20	1	0	138
421	FICTICIA C	1	0	138
422	FICTICIA D	1	0	138
423	TRAFO 1	1	0	230
424	TRAFO 2	1	0	230
425	TRAFO 3	1	0	138
426	TRAFO 4	1	0	230
427	TRAFO 5	1	0	138
428	TRAFO 6	1	0	500
429	TRAFO 7	1	0	230
430	TRAFO 8	1	0	500
431	TRAFO 9	1.	0	138
432	TRAFO 10	1	0	230
433	TRAFO 11	1	0	500
434	TRAFO 12	1	0	230

Os nomes dados aos barramentos visam explicitar os circuitos a eles conectados. Os barramentos fictícios identificam as barras desconectadas da rede que serão integradas nos apropriados anos de planejamento, quando for alterado o valor de geração e carga para as mesmas. No exemplo, há três classes de tensão, 500 kV, 230 kV e 138 kV. A barra de referência é o barramento 404, "USINA 4".

O Quadro 8.2 contém dados referentes aos transformadores existentes na configuração básica, presentes no arquivo do caso base.

QUADRO 8.2 - DADOS DE TRANSFORMADORES EXISTENTES

BARRA INICIAL	BARRA FINAL	NÚMERO DE CIRCUITOS EXISTENTES	FAIXA DE PASSAGEM	CAPACIDADE SOB COND. NORMAIS (MVA)
401	423	1	2	400
402	424	1	2	400
409	426	1	2	200
403	427	1	2	200
411	428	1	2	400
412	429	1	2	400
430	416	1	2	400
413	433	1	2	400

O Quadro 8.3 contém dados referentes às linhas de transmissão existentes no configuração básica, presentes no arquivo do caso base.

QUADRO 8.3 - DADOS DE LINHAS DE TRANSMISSÃO EXISTENTES

BARRA INICIAL	BARRA FINAL	NUMERO DECIRCUITOS EXISTENTES	FAIXA DE PASSAGEM	CAPACIDADE SOB COND. NORMAIS (MVA)
401	402	1	2	1780
424	403	1	2	400
403	411	1	2	400
404	428	1	2	1780
404	412	1	2	1780
426	410	1	2	400
410	411	1	2	400
429	413	1	2	400
433	414	1	2	1780
414	415	1	2	1780
414	430	1	2	1780
423	417	1	2	400
427	405	1	2	400

Dados como custo dos equipamentos, comprimento de linhas de transmissão, bem como valores de resistências e reatâncias, não foram considerados relevantes para o presente estágio de evolução do protótipo, uma vez que são utilizados apenas pelo motor de simulação, o programa SINTRA modificado. Objetivou-se tratar com aqueles dados que explicitassem os circuitos presentes e a topologia da rede em estudo.

O Quadro 8.4 contém os valores de carga e geração, expressos em MVA, para os cinco estágios de planejamento do caso teste, presentes no arquivo do caso base.

QUADRO 8.4 - DADOS DE GERAÇÃO E CARGA PARA OS VÁRIOS ESTÁGIOS DE PLANEJAMENTO

Nº DA BARRA	ESTÁGIOS DE PLANEJAMENTO - ANOS DE ESTUDO									
	1993		1994		1995		1996		1997	
	Geração	Carga	Geração	Carga	Geração	Carga	Geração	Carga	Geração	Carga
401	258,54	0,0	293,68	0,0	298,79	0,0	279,93	0,0	298,19	0,0
402	60,53	0,0	68,53	0,0	69,72	0,0	65,32	0,0	69,58	0,0
403	215,45	0,0	244,74	0,0	249,0	0,0	233,28	0,0	248,49	0,0
404	280,09	0,0	318,16	0,0	323,69	0,0	303,26	0,0	323,04	0,0
405	89,18	0,0	97,89	0,0	298,79	0,0	279,93	0,0	298,19	0,0
407	0,0	0,0	0,0	0,0	0,0	0,0	233,28	0,0	248,49	0,0
409	0,0	115,0	0,0	120,0	0,0	125,0	0,0	130,0	0,0	135,0
410	0,0	11,0	0,0	60,0	0,0	103,0	0,0	120,0	0,0	125,0
411	0,0	187,6	0,0	210,0	0,0	250,0	0,0	280,0	0,0	280,0
412	0,0	69,5	0,0	68,0	0,0	100,0	0,0	150,0	0,0	150,0
413	0,0	89,0	0,0	100,0	0,0	130,0	0,0	135,0	0,0	136,0
414	0,0	73,8	0,0	85,0	0,0	115,0	0,0	120,0	0,0	122,0
415	0,0	186,6	0,0	205,0	0,0	210,0	0,0	210,0	0,0	218,0
416	0,0	144,0	0,0	150,0	0,0	160,0	0,0	180,0	0,0	205,0
417	0,0	24,1	0,0	25,0	0,0	40,0	0,0	60,0	0,0	100,0
420	0,0	0,0	0,0	0,0	0,0	7,0	0,0	10,0	0,0	15,0
TOTAIS	*903,79*	*900,60*	*1023,00*	*1023,00*	*1239,99*	*1240,00*	*1395,00*	*1395,00*	*1485,98*	*1486,00*

Como mostra este Quadro, e a Figura 8.2, o balanço de potência é nulo para todo o período de estudo. Contudo, há estrangulamentos, sobrecargas nos ramos de transmissão como será mostrado.

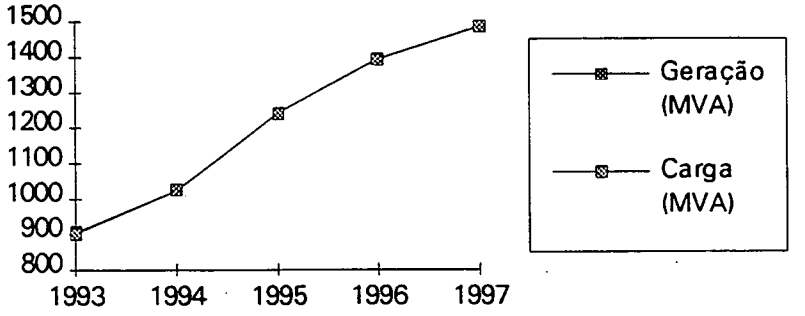


FIGURA 8.2 - DIAGRAMA DE CARGA E GERAÇÃO PARA TODO PERÍODO DE ESTUDO

8.2.2. Arquivo de adições

O arquivo de adições, ***ad-gpse.dat***, contém dados a respeito dos ramos de transmissão a expandir. Como o protótipo SAPIENS, sua última versão, foi modelado para tratar apenas com linhas de transmissão e transformadores, não foram considerados outros tipos de circuitos neste exemplo*.

O Quadro 8.5 contém dados de interesse ao protótipo SAPIENS referentes às linhas de transmissão planejadas, presentes no arquivo de adições.

QUADRO 8.5 - DADOS DE LINHAS DE TRANSMISSÃO PLANEJADAS

BARRA INICIAL	BARRA FINAL	FAIXA DE PASSAGEM	CAPACIDADE SOB COND. NORMAIS (MVA)
425	409	2	135
423	411	2	400
423	406	2	400
405	431	2	135
406	416	2	400
407	417	2	400
408	434	2	400
408	416	2	400
408	417	2	400
433	415	2	1780
432	416	2	400
423	419	2	400
411	418	2	400
420	421	2	135
420	422	2	135
409	421	2	135
425	422	2	135

* Transformadores defasores, links DC, etc.

O Quadro 8.6 contém dados referentes aos transformadores planejados, presentes no arquivo de adições.

QUADRO 8.6 - DADOS DE TRANSFORMADORES PLANEJADOS

BARRA INICIAL	BARRA FINAL	FAIXA DE PASSAGEM	CAPACIDADE SOB COND. NORMAIS (MVA)
425	402	2	200
431	411	2	200
432	404	2	400
434	415	2	400
405	418	2	200
405	419	2	200

8.3. SIMULAÇÃO DO CASO TESTE

De posse destes dados, foi feito um estudo de planejamento com o modelo SINTRA, fazendo uso das estratégias de expansão FORWARD e BACKWARD. O Quadro 8.7 apresenta os resultados obtidos, listando os circuitos adicionados e o ano de entrada dos mesmos.

QUADRO 8.7 - RESULTADOS DA SÍNTESE DA EXPANSÃO

FORWARD				BACKWARD			
Circuito Adicionado		Custo a Valor Presente	Ano Ent.	Circuito Adicionado		Custo a Valor Presente	Ano Ent.
De	Para	(US$ 10^3)		De	Para	(US$ 10^3)	
405	418	8805	1993	432	416	14027	1993
411	418	15194	1993	432	404	11104	1993
423	411	37353	1993	423	411	37353	1993
432	404	11104	1993	405	418	6152	1995
432	416	14027	1993	405	418	6152	1995
405	418	6152	1995	411	418	10615	1995
425	402	5003	1996	420	421	642	1995
425	422	1846	1996	409	421	1708	1995
420	422	522	1996	408	417	16155	1996
407	417	14118	1996	408	416	8917	1996
408	417	16155	1996	407	417	14118	1996
408	416	8917	1996				
Custo Total		139196		Custo Total		126943	

Um aspecto a ressaltar neste Quadro é a economia trazida pela estratégia BACKWARD em comparação com o custo total da estratégia FORWARD. Para referências correlatas, ver [8].

Não foi preciso fazer uso da síntese automática para eliminar os problemas de sobrecarga de potência ativa nos ramos de transmissão. Buscou-se apenas duplicar os circuitos sobrecarregados, buscar novos caminhos para escoamento do fluxo de potência e integrar ao sistema as barras que estivessem desconectadas e com injeção líquida de potência ativa diferente de zero.

O diagrama unifilar desta rede foi desenhado em programa gráfico para Microsoft Windows, o RFFLOW, e usado (após impresso) como elemento de apoio no processo de síntese da expansão (vide Figura 3). A visualização da topologia da rede permitiu acompanhar o fluxo na rede nos momentos em que o mecanismo de adição automática não propunha reforços ao sistema de transmissão.

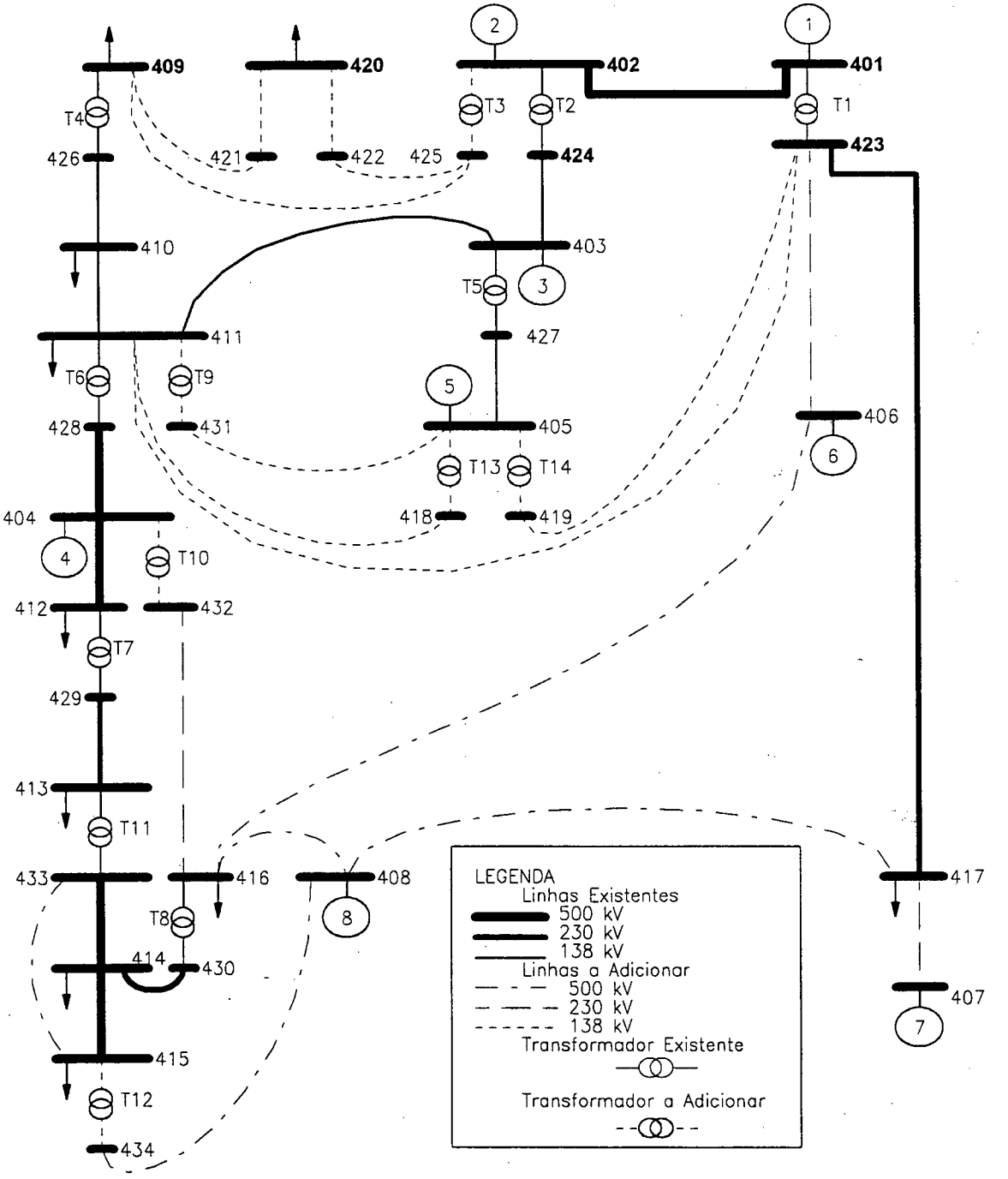


FIGURA 8.3 - DIAGRAMA UNIFILAR DO SISTEMA TESTE

8.4. DESEMPENHO DO PROTÓTIPO SAPIENS

Este estudo de síntese preliminar serviu de referência nos testes que se seguiram com o protótipo SAPIENS. A este submetido o mesmo caso aqui descrito e realizado a mesma sequência de adições de circuitos. Os comandos foram passados conforme descrito pelo Capítulo 7. Os resultados finais foram os mesmos apresentados no Quadro 8.7, concluindo-se pela validade do protótipo como modelo experimental.

Como vantagens do protótipo SAPIENS sobre o modelo SINTRA, versão de Set/92, notou-se: a interface gráfica do protótipo, em contraste com o uso de papel e lápis durante o processo de síntese em seção de trabalho com o modelo SINTRA; o diagnóstico do sistema alertando para a presença de barras isoladas e linhas sobrecarregadas*. O próprio diagrama unifilar indicava o caminho do fluxo nos casos de sobrecarga, sendo mais direta a tarefa de propor soluções a este tipo de problema.

8.4.1. Tempo de processamento

Não se pode falar em tempo de processamento quando se está lidando com sistemas interativos. Tais sistemas, por permitirem o emprego de processos manuais, por interagirem com o usuário em tempo de execução, aguardam até que algum comando lhes seja solitado. Quanto à execução deste comando, é plausível falar em tempo de processamento, uma vez que são acionados mecanismos automáticos, modelados para traduzir a tarefa a executar.

Os comandos acionados pelo protótipo SAPIENS são os mesmos empregados pelo modelo SINTRA*. Na verdade, como já foi dito em capítulos anteriores, o protótipo SAPIENS utiliza o

---

* Este diagnóstico, como foi colocado no Capítulo 7, é dado de duas formas, graficamente, em códigos de cores sobre o diagrama unifilar, e por escrito, na forma de mensagens ao usuário.

* Trata-se aqui, dos comandos de análise, síntese e gerais [7].

modelo SINTRA modificado como seu motor de simulação. Logo, não se pode dizer qual dos dois tem menor tempo de processamento.

Outrossim, há alguns detalhes a considerar. O protótipo SAPIENS está ambientado no sistema Windows, enquanto o modelo SINTRA é um programa para DOS. Para simular os comandos de análise, síntese e gerais, o protótipo SAPIENS provoca uma interrupção no sistema Windows que passa, temporariamente, o controle para o DOS, onde é executado o programa SINTRA. Tal procedimento é comum quando se lida com programas ambientados em sistemas diferentes, o que torna o protótipo SAPIENS aparentemente mais lento do que o programa SINTRA.

Contudo, isto é temporário e se deve apenas ao software base disponível, com o qual se compilou o programa SINTRA* (vide Capítulo 6). Basta compilar o motor de simulação com uma linguagem voltada para Windows (vide Capítulo 6) que este fato deixa de existir.

Por fim, melhoramentos ainda devem ser feitos ao protótipo, passando-o a um modelo de pesquisa. Caso se deseje continuar com o modelo SINTRA modificado como motor de simulação, esforços iniciais podem se concentrar no trato dos comandos de análise, tornado-os mais interativos e ergonômicos. Os comandos gerais e de síntese necessitam apenas do mouse, um dispositvo apontador, para serem operados. Os comandos de análise ainda demandam o uso do teclado, de forma reduzida, porque o usuário apenas necessita indicar os parâmetros finais dos mesmos, ficando o resto a cargo do SAPIENS (vide Figura 8.4). Estes comandos ainda descendem da segunda versão do protótipo SAPIENS, quando, por opção, usou-se o inglês na interface com o usuário.

---

* FORTRAN 77 LAHEY, versão DOS.

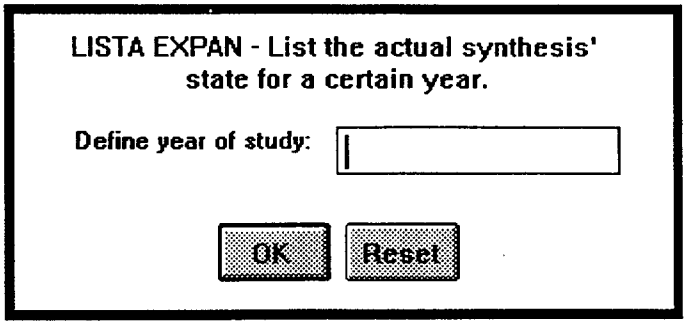


FIGURA 8.4 - JANELA DE DIÁLOGO PARA COMANDOS DE ANÁLISE

8.5. CONCLUSÕES

A metodologia apresentada nesta dissertação orienta a formação de um sistema baseado no conhecimento de uso comercial, no entanto, sua aplicação prática ficou limitada à criação do modelo experimental, como ponto de partida para um sistema de produção.

O presente capítulo evidencia a validade do protótipo SAPIENS como modelo experimental de apoio ao processo decisório de planejamento da expansão, com base em comparações com os resultados oferecidos por outro modelo interativo, o modelo SINTRA, este último em uso pela ELETROBRÁS. O desempenho apresentado pelo protótipo SAPIENS pode ser considerado satisfatório frente às limitações técnicas encontradas durante a fase de implementação. Tais limitações são temporárias, não sendo entraves à sua evolução, como se procurou esclarecer no presente capítulo.

O protótipo SAPIENS é bastante flexível e praticamente não está limitado a nada, nem mesmo ao motor de simulação, que poderá vir a ser modelado como um sistema orientado a objeto [53].

O próximo capítulo ressalta algumas recomendações à equipe que se engajar em futuras pesquisas, bem como algumas contribuições provenientes da presente dissertação.

CAPÍTULO 9- CONCLUSÕES E RECOMENDAÇÕES

9.1. JUSTIFICATIVA PARA O TRABALHO

Nos decorrer da presente exposição, procurou-se mostrar como teoria e prática podem ser adequadamente conjugados no desenvolvimento de protótipos computacionais flexíveis e consistentes. O teor de complexidade do processo de planejamento impede a completa modelagem do mesmo (vide Capítulo 2); os emergentes paradigmas de programação orientada a objeto, juntamente com a evolução das técnicas de modelagem de sistemas baseados no conhecimento (vide Capítulos 3, 4 e 5), fornecem subsídios importantes para implementar modelos aproximados, porém, flexíveis e portáveis.

Em vista disso, com o emprego da metodologia apresentada no Capítulo 6, foi implementado o protótipo SAPIENS, em versões sucessivamente melhoradas, o que ressalta a sua flexibilidade como modelo computacional (vide Capítulo 7).

A seguir, algumas contribuições desta dissertação.

9.2. CONTRIBUIÇÕES

Como contribuições, este trabalho cita:

a) Levantamento bibliográfico sobre aplicações de inteligência artificial ao processo de planejamento de sistemas de potência;

b) Estudo prático com o modelo SINTRA, ELETROBRÁS, com o sistema Sul e Sudeste brasileiro, com os quais foi possível comprovar economia de escala com o emprego da estratégia BACKWARD;

c) Estudo sobre o código fonte do modelo SINTRA, levantando os algoritmos da Fase I e Fase II, na forma de fluxogramas, e gerando uma versão deste modelo capaz de interpretar os comandos através de um arquivo pré-estabelecido;

d) Modelagem e implementação de um sistema baseado no conhecimento para apoio ao processo decisório do planejamento da transmissão;

e) Apresentação de uma metodologia para desenvolvimento de sistemas baseados no conhecimento;

f) Estudo e implementação de DLL (Dynamic Link Library), para trabalhar com o protótipo SAPIENS.

A seguir, algumas recomendações para futuros tópicos de pesquisa.

9.3. RECOMENDAÇÕES PARA FUTUROS TRABALHOS

No que diz respeito à síntese automática, alguns melhoramentos podem ser incluídos:

a) Criar uma DLL para ler o arquivo de resultados com o diagnóstico do sistema. Após cada adição, seja ela automática ou manual, o SINTRA emite um relatório sobre o impacto desta medida sobre o estado anterior da rede em estudo. Este relatório permite identificar os circuitos adicionados e/ou retirados pela síntese automática. Isto será útil para se mesclar com o diagrama unifilar na interface gráfica, permitindo ao usuário acompanhar o processo diretamente na tela;

b) Incluir motores de simulação alternativos, tais como um programa de fluxo de potência convencional, complementando o presente algoritmo de simulação, qual seja, o modelo de fluxo DC. Seria conveniente, neste caso, adotar uma linguagem de programação adequada ao ambiente Windows, eliminando a necessidade de *shells* para DOS;

c) Efetivar estudos com linguagem C++ para Windows. Isto abre as portas para sanar as limitações da atual ferramenta de desenvolvimento, o KAPPA-PC*. As rotinas para dispositivos apontadores, quando incluidas ao projeto, eliminariam os *ComboBox*, permitindo a seleção dos circuitos no próprio diagrama unifilar. Isto também daria a oportunidade de testar o código fonte do protótipo em ambiente UNIX, multiusuário (estações de trabalho);

d) Desenvolver métodos para criação de caminhos para expansão. Isto seria útil após detectar-se barras isoladas. Como primeira medida, o sistema emitiria uma lista contendo os circuitos que compõem o caminho que integraria esta barra à rede. O usuário poderia, ou não, usar este caminho em seu plano de expansão;

e) Com o emprego de DDE (*Dynamic Data Exchange* - não explorada conceitualmente por esta dissertação), estaria aberta a possibilidade de conexão a uma base de dados, ao invés de usar arquivos ASCII, facilitando o intercâmbio de informações entre SAPIENS e demais aplicativos for Windows;

f) Desenvolver um sistema integrado, multidisciplinar, de análise e suporte à decisão, a partir da infra-estrutura implementada no protótipo SAPIENS, conforme esquematizado pela Figura 9.1.

---

* Padrões de cores, novas rotinas pré-definidas para lidar com janelas e dispositivos apontadores (mouse, trackballs, canetas óticas, etc.).

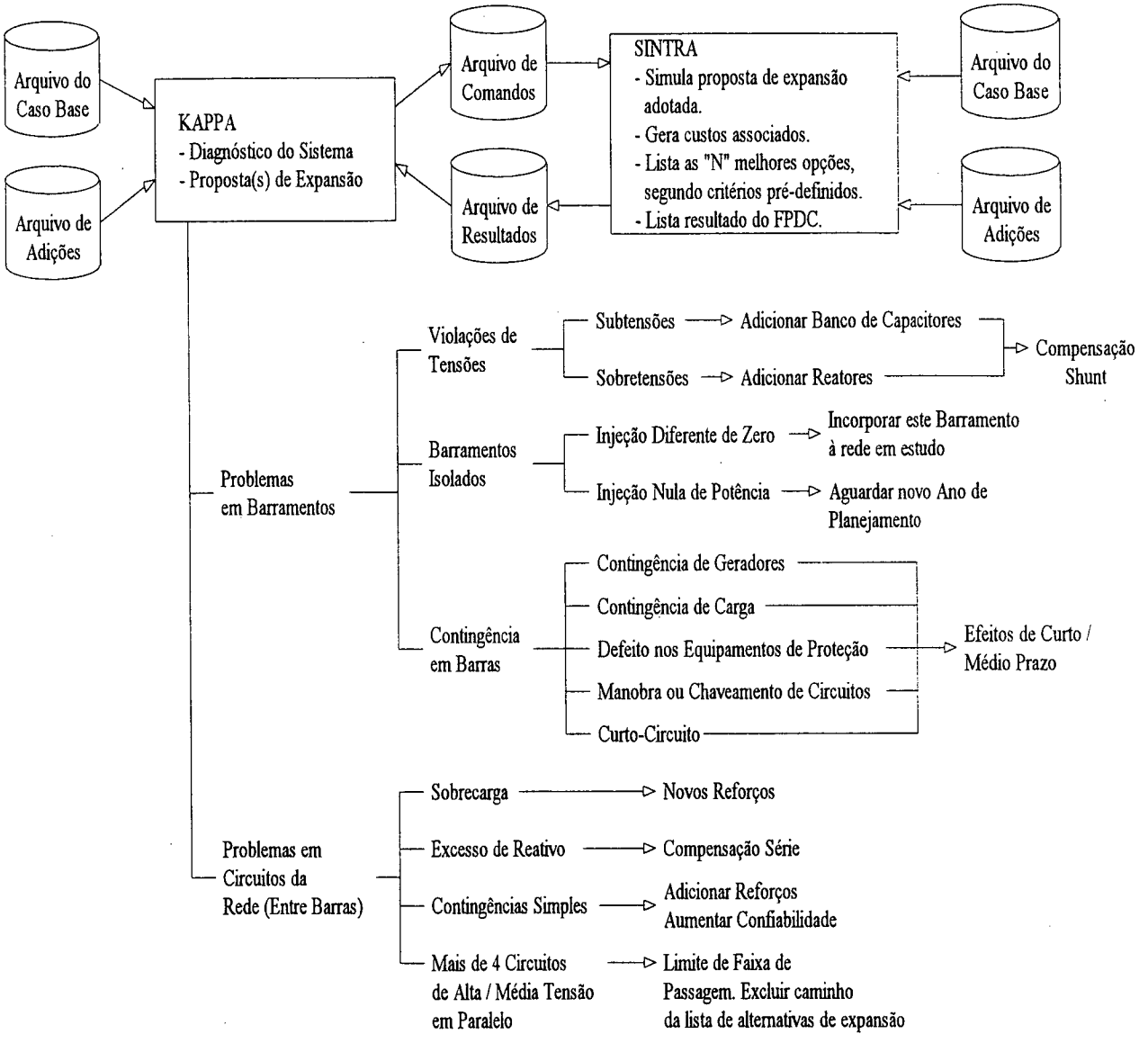


FIGURA 9.1 - SAPIENS / VERSÃO MULTIDISCIPLINAR

Este sistema contaria com as seguintes características:

- Neste sistema, a integração continuaria sendo através de arquivos de dados e arquivos resultados. O SAPIENS faria inferências na base de fatos da rede de transmissão em estudo, a fim de encontrar estados que possam corromper a continuidade do fornecimento de energia. Em seguida, apresentaria ao

planejador o resultado deste diagnóstico, sugerindo-lhe métodos de análise adequados a cada situação;

- As heurísticas são priorizadas. Assim, na Figura 9.1, a busca por contingências de circuitos é mais prioritária do que a busca por contingências de barramentos. Porém, a incorporação dos barramentos isolados é mais importante do que a análise de sobrecarga, tendo em vista que a integração destes barramentos ao sistema, pode mudar o sentido do fluxo na rede.

9.4. CONCLUSÕES FINAIS

Estas são apenas algumas sugestões. A abrangência das mesmas dependerá do empenho e profundidade dados pela equipe de desenvolvimento, bem como o suporte disponível a nível de hardware e software.

O protótipo SAPIENS está pronto. Espera-se pois, que outras equipes de trabalho o encarem, agora do ponto de vista de um sistema de pesquisa (vide Capítulo 6), gerando um modelo de uso prático e versátil, a partir do estado atual em que este protótipo se encontra. Este processo evolutivo conduzirá a outros ramos de investigação e desenvolvimento, que, tomando-se os devidos cuidadosfomentando as pesquisas em novas áreas.

Investir em tecnologia é a melhor política para o desenvolvimento. Os custos de investimento em *R&D*, mesmo somados e atualizados, não podem superar os lucros futuros.

APÊNDICE A - O MODELO SINTRA

A.1. MODELO DE FLUXO DE CARGA LINEARIZADO

Conforme [7], uma configuração da rede elétrica é considerada promissora na expansão da transmissão quando atende certas condições de custo e limites de segurança. Nos estudos de longo prazo, estes limites dizem respeito principalmente às condições de carregamento nos ramos, as quais devem ser verificadas identificando-se os fluxos nas ligações e comparando seus valores com os limites impostos.

O modelo de fluxo de carga linearizado que esta seção irá apresentar tem a característica de baixo custo computacional aliada a uma precisão apropriada para o estudo do planejamento da expansão, sendo bastante usado neste setor.

A.1.1. Representação matemática dos componentes da rede

Num modelo de fluxo de carga, os elementos da rede elétrica são representados por intermédio de seus parâmetros elétricos característicos como, por exemplo, resistência, reatância e admitância shunt para as linhas de transmissão, impedância para os transformadores, etc.

A seguir, é apresentado o modelo π equivalente, representando uma linha de transmissão genérica (Figura A.2). Os parâmetros $\mathbf{r_{ij}}$, $\mathbf{x_{ij}}$, $\mathbf{bsh_{ij}}$, $\mathbf{I_{ij}}$, $\mathbf{V_i}$ e θ_i, mostrados na figura são, respectivamente, resistência, reatância, susceptância shunt e corrente da linha i-j, e a magnitude de defasagem da tensão na barra i.

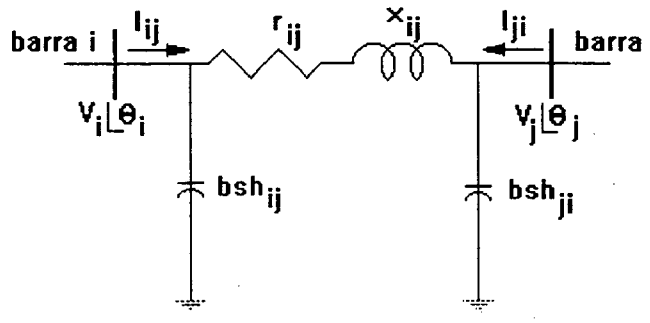


Figura A.1 - Modelo π de uma linha de transmissão

Aplicando as Leis de Kirchoff e as restrições operacionais do sistema neste tipo de representação, um conjunto de equações e inequações não-lineares envolvendo as grandezas elétricas pode ser obtido para o cálculo de fluxo de carga. A dedução das equações, não sendo do escopo deste trabalho, pode ser encontrada em [18]. A seguir, o texto procurará apresentar o conjunto de equações e inequações não-lineares gerais. Mais adiante, através de algumas simplificações, será então mostrado o modelo linearizado.

A.1.2. Equações gerais de fluxo de carga

As equações não-llineares empregadas no cálculo de fluxos de potência da rede em regime permanente são:

$$P_i = V_i \cdot \sum_{j \in \Omega_i}^{N_B} V_j \cdot (G_{ij} \cdot \cos\theta_{ij} - B_{ij} \cdot \operatorname{sen}\theta_{ij}) \qquad \text{(A.1)}$$

$$Q_i = V_i \cdot \sum_{j \in \Omega_i}^{N_B} V_j \cdot (G_{ij} \cdot \operatorname{sen}\theta_{ij} + B_{ij} \cdot \cos\theta_{ij}) \qquad \text{(A.2)}$$

onde:

N_B - número total de barras existentes;

P_i - injeção líquida de potência ativa na barra i;

Q_i - injeção líquida de potência reativa na barra i;

V_i - tensão da barra i;

Ω_i - conjunto de barras adjacentes à barra i ligadas a ela através de algum ramo;

θ_i - abertura angular da tensões no ramo i-j ($\theta_{ij} = \theta_i - \theta_j$);

G_{ij}, B_{ij} - respectivamente, parte real e imaginária do elemento Y_{ij} da matriz Y de admitância nodal ($Y_{ij} = G_{ij} + j.B_{ij}$).

Os elementos Y_{ij} estão em função da resistência, reatância e susceptância shunt das linhas de transmissão e dos transformadores:

$$Y_{ij} = \frac{r_{ij}}{r_{ij}^2 + x_{ij}^2} - j \cdot \frac{x_{ij}}{r_{ij}^2 + x_{ij}^2}, \; i \neq j \tag{A.3}$$

$$Y_{ij} = \frac{1}{Z_{ij}} \tag{A.4}$$

$$Y_{ii} = j \cdot bsh_i + \sum_{j \in \Omega_i}^{N_B} (j \cdot bsh_{ij} + Y_{ij}) \tag{A.5}$$

No cálculo de fluxos de carga, consideram-se três tipos de barras: barras de tensão controlada (barras PV), barras de carga (barras PQ) e a barra de referência (barra Vθ), sendo que a cada tipo de barra é associado um conjunto distinto de restrições:

As restrições genéricas existentes são:

$$V_i^{min} \leq V_i \leq V_i^{max} \tag{A.6}$$

$$P_i^{min} \leq P_i \leq P_i^{max} \tag{A.7}$$

$$Q_i^{min} \leq Q_i \leq Q_i^{max} \tag{A.8}$$

Para as barras de tensão controlada, a serem consideradas no modelo de fluxo de carga linearizado, as restrições usadas são (A.7) e (A.8).

A.1.3. Equações linearizadas

O modelo de fluxo de carga linearizado, ou de fluxo DC, é obtido através da aplicação das seguintes hipóteses simplificadoras às equações (A.1) e (A.2):

a) considera-se conhecidos e iguais a 1 p.u. (normalização - por unidade) os módulos de tensão de todas as barras do sistema;

b) a reatância de uma linha de transmissão é muito maior que a sua resistência ($x_{ij} >> r_{ij}$);

c) as aberturas angulares entre barras são suficientemente pequenas, tais que, $sen\theta_{ij} \approx \theta_{ij}$;

A equação do modelo linearizado assim obtido é:

$$P_i = -\sum_{j \in \Omega_i}^{N_B} B_{ij} \cdot \theta_{ij} \tag{A.9}$$

sendo que,

$$B_{ij} \cong \frac{1}{x_{ij}} \tag{A.10}$$

O conjunto das equações (A.9) e (A.10), numa forma matricial, corresponde a:

$$B \cdot \Theta = P \tag{A.11}$$

onde:

P - vetor de injeções de potência ativa P_i;

B - matriz susceptância B_{ij};

Θ - vetor de ângulos de tensões θ_i.

O problema da obtenção de fluxos de potência no modelo linearizado se resume então na solução do sistema de equações lineares (A.11).

A dimensão da matriz B é N_B^2, ou seja, o quadrado do número de barras do sistema em estudo.

Para sistemas pequenos (por exemplo, até 30 barras), a solução do sistema pode ser obtida invertendo-se diretamente a matriz B [31]:

$$\Theta = B^{-1} \cdot P \qquad \text{(A.12)}$$

Entretanto, nos sistemas de grande porte (acima de 200 barras, por exemplo), torna-se impraticável, em termos computacionais, a utilização da matriz B^{-1}, uma vez que se trata de uma matriz normalmente cheia, ao contrário da matriz B que é bastante esparsa. Esta particularidade da matriz B torna conveniente a utilização de técnicas de esparsidade como em [49], onde somente os elementos não-nulos da matriz B são armazenados e manipulados.

A resolução das equações (A.11) neste tipo de sistema é feita usando métodos de solução baseados na eliminação de Gauss.

Em termos de desempenho e precisão, o modelo de fluxo de carga DC foi comparado com o modelo não-linear AC, apresentando um resultado bastante satisfatório nos estudos de longo prazo [31].

A.2. ÍNDICE DE SENSIBILIDADE POR MÍNIMO ESFORÇO

Como consta em [16], pode-se mostrar que a solução do fluxo de potência linearizado, sob o modelo da equação (A.11), corresponde a uma distribuição de fluxos no sistema, que é equivalente à solução do problema quadrático abaixo [57]:

$$\text{Min } z = \frac{1}{2}\sum_{j} x_j t_j^2 \tag{A.13}$$

$$\text{suj. a } \sum_{j\in\Omega_k}^{N_B} t_j = p_k, \forall k \tag{A.14}$$

onde:

z - índice de desempenho para uma dada configuração de rede;

x_j - impedância do ramo j;

t_j - fluxo de potência ativa no ramo j.

A solução deste problema é a mesma solução do fluxo de potência linearizado, indicando que a distribuição dos fluxos segue a lei do *mínimo esforço* que objetiva minimizar (A.13) sujeito a (A.14). Esta lei pode ser utilizada como índice de desempenho para ordenar os candidatos à adição, sendo a adição de maior impacto sobre **z** aquela mais favorável para a distribuição *natural* dos fluxos. A sensibilidade do índice **z** em relação à susceptância y_i do ramo i é:

$$\frac{\partial Z}{\partial y_i} = -\frac{1}{2}\varphi_i^2 \tag{A.15}$$

onde:

$$\varphi_i = \Theta_k - \Theta_l \tag{A.16}$$

onde:

k, l - barras terminais do ramo i;

Θ - solução de (A.11) antes da adição do circuito i.

Note-se que o cálculo de (A.15) para qualquer circuito candidato é imediato conhecida a solução de (A.11), o que permite ordenar os candidatos por um índice benefício/custo, onde o benefício Δz_i da adição do circuito i é estimado por:

$$\Delta z_i = -\frac{1}{2}\varphi_i^2 \cdot \Delta y_i \qquad \text{(A.17)}$$

onde:

Δy_i - susceptância série do circuito i.

Após fazer esta ordenação para cada classe de tensão, os melhores candidatos por classe irão compor os circuitos iniciais dos chamados caminhos de mínimo esforço. Se as barras terminais de um circuito não forem barras intermediárias (sem injeção), o caminho está formado. Caso contrário, de cada barra intermediária busca-se o melhor candidato pelo mesmo critério, repetindo-se este procedimento até não haver barras intermediárias. Completado um caminho, pode-se simular por compensação o efeito (redução de sobrecarga na rede) devido a sua adição e somando-se os custos de cada circuito resulta o custo do caminho [57].

A relação benefício/custo de cada caminho por classe de tensão pode ser utilizada para selecionar o caminho mais atraente a ser adicionado à rede sobrecarregada.

A sensibilidade Δz_i pode também ser utilizada para o reforço da rede para critério de contingências simples, definindo-se este índice como o maior valor encontrado durante a análise de contingências.

A.3. O PROCESSO DE SÍNTESE NO MODELO SINTRA

A síntese automática realizada pelo SINTRA pode ser dividida em duas fases distintas. Na Fase 1 (Figura A.2), o programa SINTRA efetua adições na rede de modo a eliminar todas as sobrecargas existentes no sistema de transmissão, o que corresponde à tarefa de *Análise do Sistema em Regime Permanente* com estudos de *Fluxo de Potência em Condições Normais* (vide Capítulo 2). A Fase 2 (Figura A.3) é responsável pela implantação de reforços adicionais no sistema de transmissão para proteção em contingências simples, o que corresponde à *Análise do Sistema em Regime Permanente*, porém com estudos de *Fluxo de Potência em Condições de Emergência* (vide Capítulo 2).

Na Fase 1 (Figura A.2), a rede é expandida automaticamente até que toda a sobrecarga tenha sido eliminada e todas as barras com injeções diferentes de zero tenham sido conectadas [50]. O critério de reforço é baseado na relação benefício/custo, sendo o benefício avaliado através da redução das sobrecargas do sistema produzida pela adição de um ou mais circuitos (caminhos).

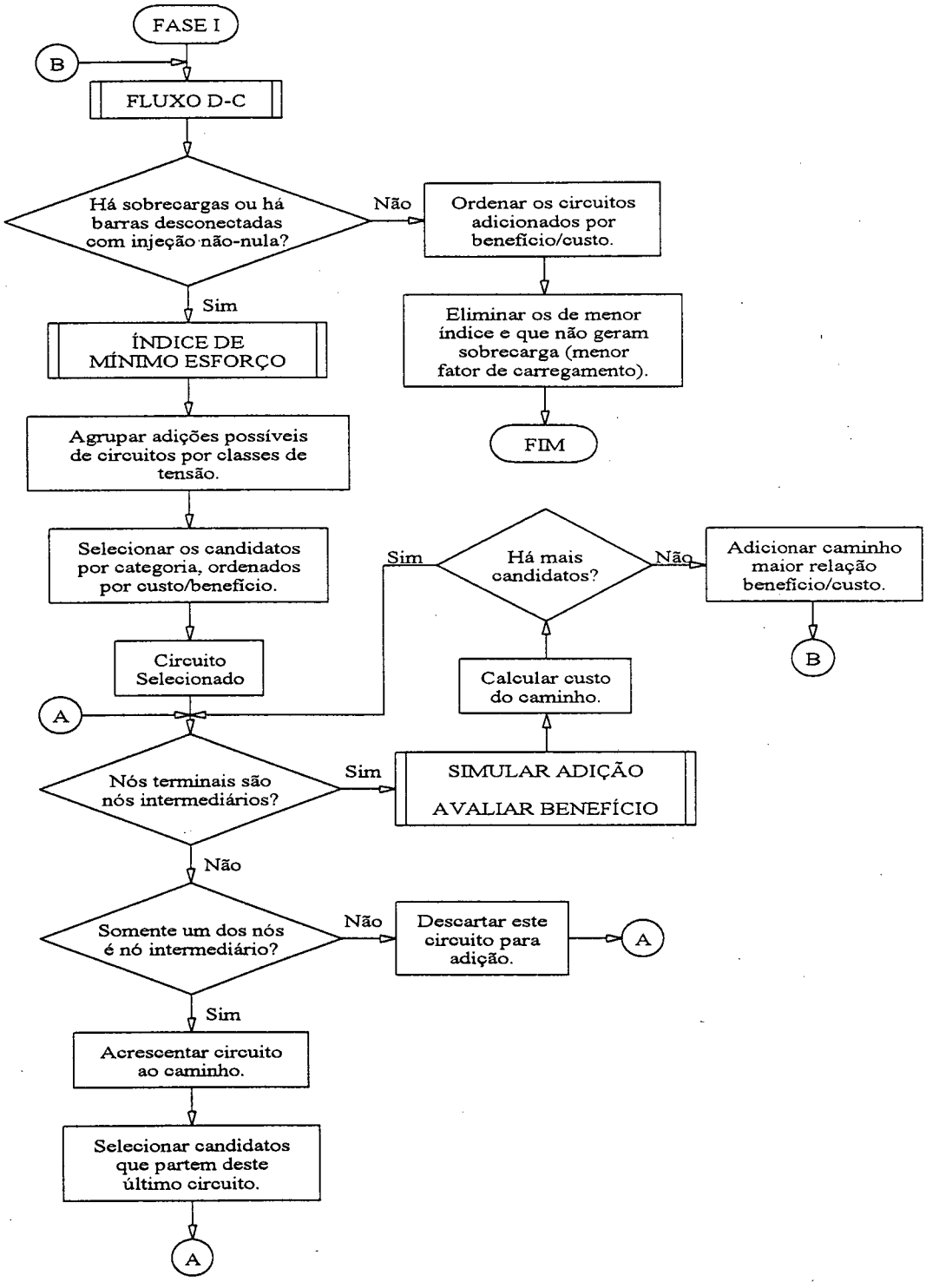


Figura A.2 - Fluxograma do SINTRA / fase I

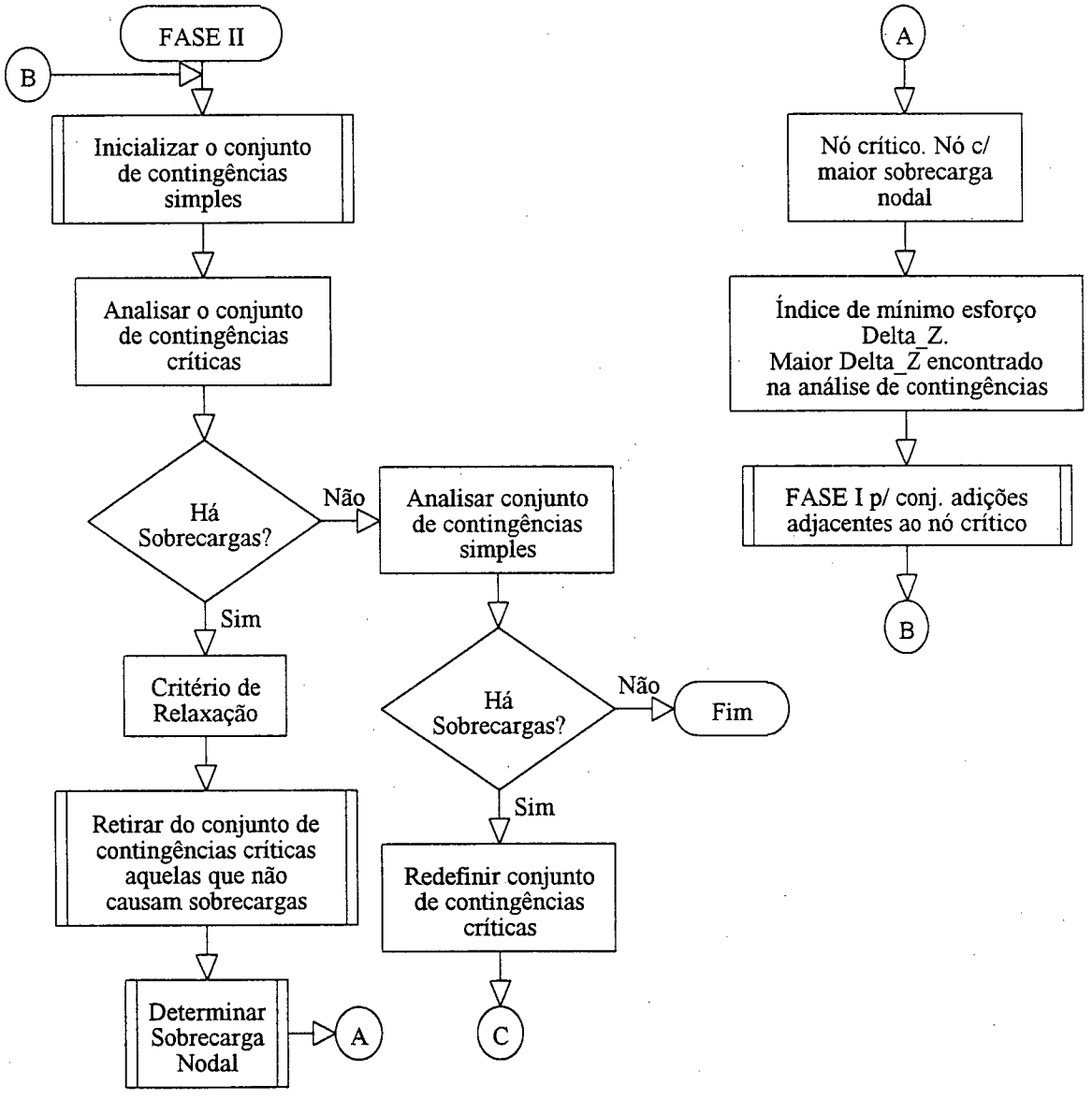


Figura A.3- Fluxograma do SINTRA / fase II

Na Fase 2 (Figura A.3), são simuladas contingências simples em todos os N circuitos da rede (*critério N-1*) e forma-se uma lista com os circuitos que provocam sobrecarga (*subconjunto crítico*), calculando-se em seguida a barra de maior *sobrecarga nodal** (*barra crítica*) [50]. Em seguida, o programa forma uma lista de caminhos candidatos à adição de forma similar à descrita

* Sobrecarga nodal - soma algébrica das sobrecargas provocadas pelos fluxos que entram na barra (positivos) e dos que saem da barra (negativos).

na Fase 1, com todos os circuitos que partem da barra crítica e todos os circuitos do subconjunto crítico, obtendo para cada um o índice de custo/benefício relacionado à adição do caminho, adicionando à rede o de melhor índice.

No caso do modelo SINTRA, o processo de síntese de redes de transmissão utilizado está representado na Figura A.4.

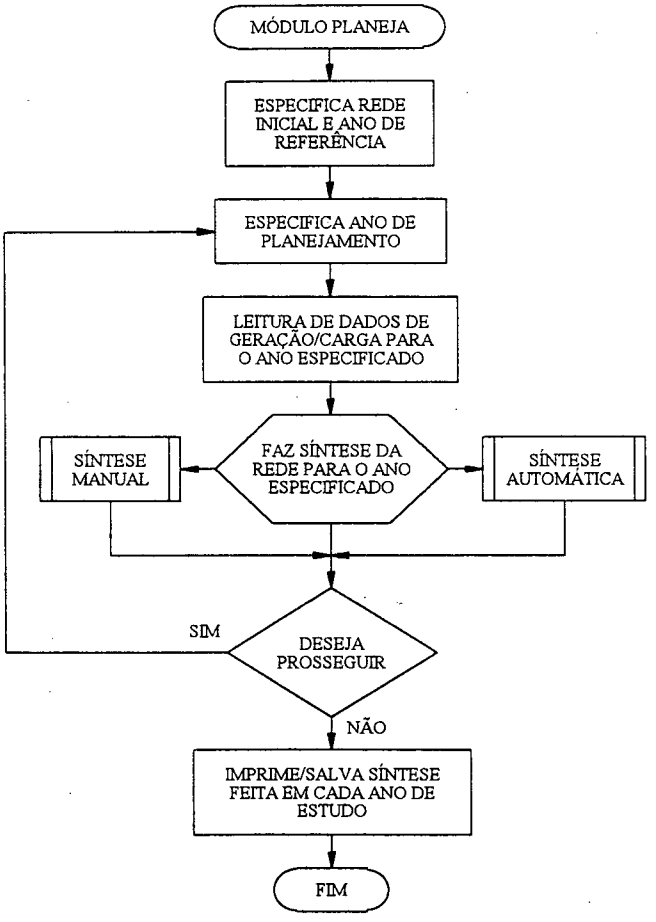


Figura A.4 - Fluxograma do SINTRA (módulo PLANEJA)

No modelo SINTRA, o planejador pode usar suas próprias heurísticas (síntese manual) quando a síntese automática se mostrar ineficaz para o problema em questão*.

---

* Uma limitação, ou exceção, dos critérios de análise e síntese adotados

APÊNDICE B - ESTRATÉGIAS DE PLANEJAMENTO ADOTADAS PELA ELETROSUL

B.1. INTRODUÇÃO

Este apêndice apresenta algumas estratégias de Planejamento da Expansão de Sistemas adotadas pela ELETROSUL. Tais estratégias constam do *ESTUDO DE PLANEJAMENTO* sob o título, *Estudo de Integração das Usinas Hidrelétricas de Capanema, Salto Caxias e Ilha Grande.* Serão ressaltadas as estratégias em si, não havendo preocupação de analisar os resultados atingidos pelos trabalhos aqui citados.

- Ressalta-se, primeiramente, a alocação do parque gerador e a situação e especificação de subestações de interesse (demanda). Os estudos visam analisar o desempenho do sistema frente a mudanças nessas configurações, a médio e longo prazo.

- Em seguida, define-se o ano crítico para dar início ao processo de planejamento, ou seja, aquele ano onde se sabe que haverá necessidade de reforço ao sistema. Isto pode ser obtido dos dados de geração e carga, e situar como ano crítico aquele que apresenta a maior variação percentual a partir dos anos anteriores.

- O ano horizonte fica a cargo do planejador, tendo-se o devido cuidado de se associar, ano a ano, os programas de geração. Para validar as alternativas mais promissoras no horizonte de médio prazo, simula-se seu comportamento para um período posterior ao horizonte de estudo.

- Os dados básico de mercado, geração e configuração, são aqueles dos Planos Decenais de Tansmissão e Geração.

- Nos despachos utilizados, as usinas hidráulicas são consideradas com 10% de reserva girante e as térmicas foram despachadas em sua capacidade plena (deduzidos os seus consumos próprios); além disso, no ano de entrada em operação de alguma unidade térmica, a mesma é considerada indisponível naquele ano. Os despachos são limitados ao intercâmbio máximo entre regiões, determinados pelo GTIN. Quando este intercâmbio é violado, procede-se a uma redução proporcional nas usinas distantes eletricamente da região de interesse do estudo.

- São realizados estudos de carga pesada e carga leve, mas não para todos os anos do horizonte de estudo. Os despachos das usinas hidráulicas, em carga leve, são reduzidos proporcionalmente ao mercado em cada subsistema.

Os critérios de desempenho técnico e econômico são aqueles fornecidos pelo GCPS. Para quantificação econômica das alternativas são utilizados custos modulares fornecidos pelo GCPS, referidos a um ano base.

As alternativas são analisadas estritamente sob os aspectos técnicos e econômicos, não sendo considerada a propriedade das obras.

Após entrevistas realizadas junto ao Engenheiro Eletricista César Ghisi Silvestre, foram ressaltados os seguintes pontos básicos para se medir a confiabilidade de uma dada rede elétrica (para a região Sul):

1. No máximo 4 circuitos paralelos para classes de tensões de 230 kV e 500 kV. Além deste patamar, busca-se o recondutoramento de circuitos pré-existentes.

2. Usar a distância como critério decisório quando for necessário integrar geração e carga.

3. A necessidade de potência reativa no sistema é identificada pela análise do perfil de tensão.

4. No planejamento anual, adota-se a estratégia BACKWARD, partindo-se de um ano crítico, que reflete um marcante crescimento do mercado.

5. Despacho econômico para alocar geração.

6. Fluxo DC para verificar sobrecargas.

7. Atender situações normais e de emergência.

8. Análise de Contingência.

9. Análise de Curto-Circuito.

10. Análise de Estabilidade Transitória.

Do que se pode inferir destas entrevistas, não existem regras rígidas a seguir no planejamento de um sistema elétrico. Muitas vezes, o planejamento da operação se mescla com o planejamento da expansão como árbitro na tomada de decisão por novos reforços à rede. Existem, isso sim, problemas básicos com os quais o planejador deve lidar. São eles:

B.2. BALANÇO DE POTÊNCIA.

Nesse caso, a rede é vista como um único ponto, uma única barra equivalentada, onde se faz o somatório de todas as potências geradas pelas barras PV menos o somatório de toda a demanda das barras PQ. Às vezes, barramentos de geração podem assumir valores de carga e vice-versa. São barramentos de interconexão e usam esta tática para compor as potências de intercâmbio entre áreas. Exemplo: Caso teste sistema Sudeste.

O importante aqui é a potência líquida no sistema. Não se faz uso de algoritmos numéricos muito elaborados. Contudo, um programa de fluxo de carga pode ser útil no sentido de calcular as perdas nos ramos de transmissão.

Refazem-se estudos sobre as reservas hidrológicas e térmicas disponíveis, talvez até mesmo realocando os pontos de geração em todo o sistema elétrico.

Pode ser também que o perfil de demanda tenha sido muito otimista e previsto um mercado consumidor mais elevado do que se pode esperar. Fatores políticos e sócio-econômicos são pontos chaves para análise do mercado. Não basta apenas a análise de dados passados com vistas a uma estipulação de um mercado consumidor futuro. É preciso ter o mínimo de incertezas possível para adotar o perfil de carga usado.

B.3. INTEGRAÇÃO DE NOVAS USINAS E PONTOS DE CARGA NO SISTEMA.

Uma vez que o balanço de potência é satisfatório, é feita uma pesquisa no sistema elétrico para se encontrar barras isoladas com injeção líquida de potência diferente de zero. Ou seja, novas usinas ou pontos de carga que precisam ser integrados para um dado ano de estudo.

Após identificar-se estes barramentos, busca-se uma solução para o problema. Buscam-se novos caminhos de transmissão necessários a esta integração.

Como foi ressaltado anteriormente, usa-se, na ELETROSUL, a distância dos ramos de transmissão como critério decisório para esta empreitada. Vale ressaltar que novos caminhos nem sempre são formados apenas por um circuito. Em geral, demandam dois ou mais circuitos, dentre linhas de transmissão, transformadores, transformadores defasadores, equipamentos de proteção, links DC, etc. A distância dos ramos de transmissão assenta-se principalmente ao caracter

econômico da expansão, ou seja, aos investimentos para a implementação dos novos equipamentos.

É claro que este não é um critério rígido. Um caminho que prometa futuramente um melhor aproveitamento das opções existentes (que passe perto de outros centros de cargas existentes ou potenciais) pode ser tido como mais viável, apesar de ser mais dispendioso economicamente.

Neste estágio, o uso de programas de fluxo de potência para simular o comportamento das opções disponíveis, para medir o seu benefício, é secundário, tendo em vista que a idéia principal é integrar as barras do sistema.

B.4. ANÁLISE DE SOBRECARGA

O sistema está pronto para operar. O Balanço de potência é satisfatório, e não há mais novas usinas a integrar no sistema. Assim, resta agora simular o seu comportamento frente a um programa de fluxo de potência, DC ou AC, que refletirá o seu desempenho em condições normais de regime permanente.

Caso este programa indique que há sobrecarga no sistema como um todo, é necessário apenas uma de duas estratégias: redespacho de geração ou adição de novos reforços.

O redespacho econômico de geração trabalha em cima do perfil de mercado existente e na configuração da rede como um todo. Busca modular a geração pelas exigências de mercado presentes, talvez até mesmo fornecendo mais geração onde houver maior reserva e linhas com mais folga na sua capacidade de transmissão.

Caso o redespacho não solucione o problema, é prática nas empresas adotar como viáveis os reforços que reduzam ao menos em 50% o valor total da sobrecarga que aflige o sistema.

Tais reforços são novos caminhos, caminhos estes que diferenciam do que foi exposto anteriormente (item 2). Agora não apenas a distância (o custo do caminho) é de interesse na expansão, mas também o seu benefício associado (a sua capacidade de redução de sobrecarga).

Há situações em que apenas a análise da sobrecarga, usando modelos de fluxo de potência, é suficiente para estimar um bom estado de operação para o sistema.

B.5. Perfil de Tensão

A necessidade de reativos no sistema pode ser prevista pela análise do perfil de tensão.

O colapso de tensão, a subtensão, a sobretensão, levam a fenômenos tais como queima de equipamentos com enrolamentos (motores, por exemplo). Isto é indesejável para as concessionárias e consumidores, que tentam contornar o problema.

Mesmo o problema de sobrecarga pode estar interligado ao problema de perfil de tensão e ser sanado por este último.

Neste estágio, usa-se um fluxo de potência AC e através de ajustes nos taps de transformadores e tensão na barra de geração, busca-se otimizar o estado do sistema elétrico em estudo.

B.6. Estudo Fino

As etapas que se seguiram até aqui, formam um grupo de estudos que servem para analisar o sistema em seu regime permanente, a médio e longo prazo (período de 10 a 30 anos). Para

eventos mais recentes, deve-se abordar a rede elétrica com estudos mais finos, abordando fenômenos transitórios.

B.6.1. Curto-Circuito

Surtos de Curto-Circuito podem aparecer de três maneiras básicas: fase-fase, fase-terra, fase-neutro. É preciso dimensionar o sistema, os barramentos de forma que a corrente de curto circuito, 3 vezes maior que a corrente de sequência zero, não danifique os componentes da rede por efeito Joule.

B. 6.2. Estabilidade Transitória

Analisa o comportamento dinâmico das unidades geradoras quando de uma contingência severa na rede. Tais contingências geram fenômenos mecânicos nas pás das turbinas das unidades geradoras hidráulicas. Levam também a variações de frequência no sistema e mudança nos valores de impedância e forma de ondas de corrente e tensão.

B.6.3. Análise de Contingências

Consiste na seleção de candidatos para análise de contingências. Esta fase só é possível sob assistência de especialistas que conheçam os pontos (barramentos, circuitos) críticos do sistema e para os quais deseja estipular melhores reforços. É necessário um acurado conhecimento da rede elétrica em estudo, o que não se obtém em poucas semanas.

Para todas estas análises, existem modelos bem definidos, o que acontece é uma interdependência entre o escalonamento temporal de sua execução. Seria supérfluo fazer uma análise de contingências sem nem ao menos verificar o balanço de potência. Da mesma forma, apenas a análise de sobrecarga de potência ativa não é índice suficiente para validar uma certa topologia de rede elétrica. Todos os estudos são importantes e cada um ocorre a seu tempo, mas demandando uma prévia análise anterior para evitar redundâncias de sintomas.

O sistema especialista irá efetuar duas funções distintas:

- Diagnosticar a rede em estudo, identificando os possíveis problemas e suas causas. Como foi exposto anteriormente, haverá uma priorização quanto ao grupo de problemas a diagnosticar.

- Propor soluções, interagindo com o planejador na busca de opções de expansão.

APÊNDICE C - SAPIENS / ÚLTIMA VERSÃO

C.1. INTRODUÇÃO

Este apêndice apenas expõe a listagem do código fonte gerado pelo *shell* KAPPA-PC para a última versão do protótipo SAPIENS. Este código pode ser posteriormente compilado e transformado em código executável como cita [35]. Para tanto, é preciso ter uma versão comercial do *shell* KAPPA-PC, Intelicorp Inc., que contém um *run-time*, um programinha residente que habilita o processo de link-edição do código fonte dos módulos *.kal* gerados. Doutra forma, basta usar este código fonte e traduzi-lo para uma linguagem orientada a objeto, uma vez que sua estrutura se mostrou funcional. Assim, não estaria dependente das bibliotecas *.h* do *shell* em questão, que promovem sua completa compilação.

C.2. LISTAGEM DO PROTÓTIPO

```
/*************************************************************/
/**       ALL FUNCTIONS ARE SAVED BELOW                     **/
/*************************************************************/

    /*************************************
     **** FUNCTION: New_Study
     *************************************/
MakeFunction( New_Study, [],
  {
  PostBusy( ON, "Por favor, aguarde. Preparando nova
seção de estudo" );
  Default:Start = ON;
  ReadData( Default:Base, Default:Adic );
  SetSlotOption( Study:Present, ALLOWABLE_VALUES,
Study:Years );
  InterpretFile( Default:Coord );
  PostBusy( OFF );
  ShowImage( Button8 );
  ShowImage( Drawing1 );
  ShowImage( Text1 );
  ShowImage( Text2 );
  } );

    /*************************************
     **** FUNCTION: Reset
     *************************************/
MakeFunction( Reset, [],
  {
  General:Status = FALSE;
  Study:Status = NORMAL;
  Study:Emergency = NORMAL;
  Study:Present = NULL;
```

```
  ClearList( Synth:Commands );
  ClearList( Analy:Commands );
  ClearList( General:Commands );
  ClearList( Commands:Commands );
  HideImage( Text3 );
  } );

    /************************************
     **** FUNCTION: Start
     ************************************/
MakeFunction( Start, [],
  {
  Reset( );
  Del( );
  Clear_Window( );
  HideImage( Bitmap2 );
  ShowImage( Bitmap1 );
  Data_Files( );
  } );

    /************************************
     **** FUNCTION: Display
     ************************************/
MakeFunction( Display, [file],
  If ( General:Status #= FALSE )
    Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
    Else {
       RemoveWindowMenu( Session2 );
       ShowWindow( Session2 );
       SetWindowTitle( Session2, file );
       ClearTranscriptImage( Transcript1 );
       DisplayFile( Transcript1, file );
       } );

    /************************************
     **** FUNCTION: Work
     ************************************/
MakeFunction( Work, [],
  {
  SetPostMessageTitle( Default:Title );
  SetWindowTitle( Session1, Default:Title );
  RemoveWindowMenu( Session1 );
  RemoveWindowMenu( Session2 );
  FreezeWindow( Session2 );
  IconifyWindow( KAPPA );
  HideWindow( BROWSER );
  HideWindow( KTOOLS );
  ShowWindow( Session1 );
  Session1:RightAction = NULL;
  } );

    /************************************
     **** FUNCTION: Close
     ************************************/
MakeFunction( Close, [],
  HideWindow( Session2 ) );

    /************************************
     **** FUNCTION: End_Work
     ************************************/
MakeFunction( End_Work, [],
  {
  Reset( );
  Clear_Window( );
  Good_Bye( );
  Del( );
  HideWindow( Session1 );
  SetWindowTitle( Session1, Session1 );
  SetWindowTitle( Session2, Session2 );
  SetMenuBar( Session1 );
  ResetWindowMenu( Session1 );
  ResetWindowMenu( Session2 );
  UnfreezeWindow( Session2 );
  SetPostMessageTitle( KAPPA );
  ShowWindow( KAPPA );
  HideImage( Bitmap1 );
  Session1:RightAction = Work;
  } );

    /************************************
     **** FUNCTION: Del
     ************************************/
MakeFunction( Del, [],
  {
  ForAll [ x|PowerSystem ]
      DeleteInstance( x );
  Default:Base = NULL;
  Default:Adic = NULL;
  Default:Coord = NULL;
  Default:Start = OFF;
  HideWindow( Session2 );
  ClearList( Study:Years );
  } );

    /************************************
     **** FUNCTION: Check_Flow
     ************************************/
MakeFunction( Check_Flow, [],
  {
  SetForwardChainMode( DEPTHFIRST );
  ResetForwardChain( );
  ForAll [ x|Branch ]
      Assert( x:Flow );
  ForwardChain( NULL, Global:Check_Flow );
  If IsThereAny?( [ x | Branch ], x:Status #= OVERLOAD )
```

```
    Then SetValue( Study:Status, OVERLOAD )
    Else SetValue( Study:Status, NORMAL );
  If IsThereAny?( [ x | Branch ], x:Emergency #=
OVERLOAD )
    Then SetValue( Study:Emergency, OVERLOAD )
    Else SetValue( Study:Emergency, NORMAL );
  } );

    /************************************
     **** FUNCTION: Connect_Terminals
     ************************************/
MakeFunction( Connect_Terminals, [],
  ForAll [ x|Branch ]
      {
      SetValue( x:InitBus:Link, CONNECTED );
      SetValue( x:EndBus:Link, CONNECTED );
      } );

    /************************************
     **** FUNCTION: Bus_Status
     ************************************/
MakeFunction( Bus_Status, [],
  {
  Let [List Study:Years]
    Let [n GetElemPos( List, Study:Present )]
      {
      ForAll [ G|Generator ]
          If ( ( GetElemPos( List, G:Year )
                <= n ) And Not( Null?( Study:Present ) ) )
            Then {
              G:Connection = TRUE;
              G:Color = 1;
              G:Style = 1;
              }
            Else {
              G:Connection = FALSE;
              G:Color = 5;
              G:Style = 2;
              };
      ForAll [ L|Load ]
          If ( ( GetElemPos( List, L:Year )
                <= n ) And Not( Null?( Study:Present ) ) )
            Then {
              L:Connection = TRUE;
              L:Color = 1;
              L:Style = 1;
              }
            Else {
              L:Connection = FALSE;
              L:Color = 5;
              L:Style = 2;
              };
      };

  If ( IsThereAny?( [ G | Generator ], G:Connection #= TRUE
                    And G:Number:Connection
                    #= FALSE )
      Or IsThereAny?( [ L | Load ], L:Connection #= TRUE
                    And L:Number:Connection
                    #= FALSE ) )
    Then ShowImage( Text9 );
  ShowImage( Drawing1 );
  } );

    /************************************
     **** FUNCTION: Rule_Select
     ************************************/
MakeFunction( Rule_Select, [],
  ForAll [ plan|Planned ]
      If ( plan:InitBus:Link #= DESCONNECTED And
plan:EndBus:Link
          #= DESCONNECTED And plan:InitBus:Power #=
          0 And plan:EndBus:Power #= 0 )
        Then {
          plan:Expansion = EXCLUDED;
          PostMessage( plan # " is isolated." );
          } );

    /************************************
     **** FUNCTION: Func
     ************************************/
MakeFunction( Func, [],
  ForAll [ plan|Planned ]
      If ( ( plan:InitBus:Link #= CONNECTED ) And (
plan:EndBus:Link
                                  #=
                                  CONNECTED ) )
        Then {
          Study:Path = TRUE;
          plan:Expansion = INSERT;
          } );

    /************************************
     **** FUNCTION: Adic
     ************************************/
MakeFunction( Adic, [],
  {
  Manual:Adic:Ncir += 1;
  SendMessage( Manual:Adic, PlusCircuit );
  } );

    /************************************
     **** FUNCTION: Show_System
     ************************************/
MakeFunction( Show_System, [],
  If Not( Null?( Default:Base ) )
    Then If Not( Null?( Default:Adic ) )
```

```
Then If Not( Null?( Default:Coord ) )
    Then {
      ClearList( Drawing1:ForegroundColor );
      AppendToList( Drawing1:ForegroundColor,
        0, 0, 255 );
      ForAll [ B|Bus ]
          {
          If Not( IsAKindOf?( B, Load )
                Or IsAKindOf?( B, Generator ) )
           Then {
              SetLineStyle( B:Width, B:Style,
                B:Color );
              MoveTo( B:x1, B:y1 );
              LineTo( B:x2, B:y2 );
              SetLineStyle( 1, 1, 1 );
              MoveTo( B:x1 - 3, B:y1
                              -
                              3 );
              DrawText( B:Number );
              };
          };
      ForAll [ L|Line ]
          {
          SetLineStyle( L:Width, L:Style,
            L:Color );
          MoveTo( L:x1, L:y1 );
          LineTo( L:x2, L:y2 );
          If ( L:Broken #= TRUE )
           Then {
              LineTo( L:x3, L:y3 );
              LineTo( L:x4, L:y4 );
              };
          };
      ForAll [ L|LineP ]
          {
          SetLineStyle( L:Width, L:Style,
            L:Color );
          MoveTo( L:x1, L:y1 );
          LineTo( L:x2, L:y2 );
          If ( L:Broken #= TRUE )
           Then {
              LineTo( L:x3, L:y3 );
              LineTo( L:x4, L:y4 );
              };
          };
      ForAll [ T|Transformer ]
          {
          SetLineStyle( T:Width, T:Style,
            T:Color );
          MoveTo( T:x1, T:y1 );
          LineTo( T:x1, T:y1 + 2 );
          MoveTo( T:x1, T:y1 + 4 );
          Ellipse( 2, 2 );
          MoveTo( T:x1, T:y1 + 6 );
          Ellipse( 2, 2 );
          MoveTo( T:x1, T:y1 + 8 );
          LineTo( T:x1, T:y1 + 10 );
          };
      ForAll [ T|TransformerP ]
          {
          SetLineStyle( T:Width, T:Style,
            T:Color );
          MoveTo( T:x1, T:y1 );
          LineTo( T:x1, T:y1 + 2 );
          MoveTo( T:x1, T:y1 + 4 );
          Ellipse( 2, 2 );
          MoveTo( T:x1, T:y1 + 6 );
          Ellipse( 2, 2 );
          MoveTo( T:x1, T:y1 + 8 );
          LineTo( T:x1, T:y1 + 10 );
          };
      Drawing1:TextSize = 14;
      ClearList( Drawing1:ForegroundColor );
      AppendToList( Drawing1:ForegroundColor,
        0, 0, 0 );
      ForAll [ G|Generator ]
          {
          SetLineStyle( G:Width, G:Style,
            G:Color );
          If ( G:Orientation #= Up )
           Then {
              MoveTo( G:x1, G:y1 );
              LineTo( G:x1, G:y1 + 4 );
              MoveTo( G:x1 - 2, G:y1
                            +
                            5 );
              DrawText( " G " );
              MoveTo( G:x1, G:y1 + 6.5 );
              Ellipse( 2.5, 2.5 );
              };
          If ( G:Orientation #= Down )
           Then {
              MoveTo( G:x1, G:y1 );
              LineTo( G:x1, G:y1 - 4 );
              MoveTo( G:x1 - 2, G:y1
                            -
                            8 );
              DrawText( " G " );
              MoveTo( G:x1, G:y1 - 6.5 );
              Ellipse( 2.5, 2.5 );
              };
          };
      Drawing1:TextSize = 12;
      ForAll [ L|Load ]
          {
          SetLineStyle( L:Width, L:Style,
```

```
                L:Color );
              If ( L:Orientation #= Up )
               Then {
                  MoveTo( L:x1, L:y1 );
                  LineTo( L:x1, L:y1 - 5 );
                  LineTo( L:x1 - 0.5, L:y1
                                -
                                5 );
                  LineTo( L:x1, L:y1 - 7 );
                  MoveTo( L:x1, L:y1 - 5 );
                  LineTo( L:x1 + 0.5, L:y1
                                -
                                5 );
                  LineTo( L:x1, L:y1 - 7 );
                  };
              If ( L:Orientation #= Down )
               Then {
                  MoveTo( L:x1, L:y1 );
                  LineTo( L:x1, L:y1 + 5 );
                  LineTo( L:x1 - 0.5, L:y1
                                +
                                5 );
                  LineTo( L:x1, L:y1 + 7 );
                  MoveTo( L:x1, L:y1 + 5 );
                  LineTo( L:x1 + 0.5, L:y1
                                +
                                5 );
                  LineTo( L:x1, L:y1 + 7 );
                  };
             };
          } );

   /************************************
    **** FUNCTION: Good_Bye
    ************************************/
MakeFunction( Good_Bye, [],
  {
  If ( Default:Start = ON )
   Then {
      HideImage( Bitmap1 );
      RemoveWindowMenu( Session1 );
      ShowImage( Bitmap2 );
      PostMessage( "Fim de execução." );
      };
  } );

   /************************************
    **** FUNCTION: Cancel_Adic
    ************************************/
MakeFunction( Cancel_Adic, [],
  {
  If Manual:Adic:Overload
   Then {
      Manual:Adic:Color = 4;
      Manual:Adic:Width = 3;
      }
   Else If ( Manual:Adic:Ncir #= 0 )
       Then If IsAKindOf?( Manual:Adic, Base )
            Then ( Manual:Adic:Color = 0 )
            Else Manual:Adic:Color = 5;
  Hide_Adic( );
  ShowImage( Drawing1 );
  } );

   /************************************
    **** FUNCTION: General
    ************************************/
MakeFunction( General, [],
  ShowImage( RadioButtonGroup1 ) );

   /************************************
    **** FUNCTION: Data_Files
    ************************************/
MakeFunction( Data_Files, [],
  {
  SetValue( Default:Base, SelectFile( "Arquivo Base",
"*.dat" ) );
  If Not( Null?( Default:Base ) )
   Then {
      SetValue( Default:Adic, SelectFile( "Arquivo de
Adições",
                        "*.dat" ) );
      If Not( Null?( Default:Adic ) )
       Then {
          SetValue( Default:Coord, SelectFile( "Arquivo de
Coordenadas",
                              "*.dat" ) );
          If Not( Null?( Default:Coord ) )
           Then New_Study( );
          };
      };
  } );

   /************************************
    **** FUNCTION: Overload
    ************************************/
MakeFunction( Overload, [],
  {
  ReadFlow( Default:Res );
  ForAll [ B|Network ]
      If B:Overload
       Then {
          B:Color = 4;
          B:Width = 3;
          ShowImage( Text10 );
          }
```

```
        Else {
           B:Width = 1;
           If ( B:Ncir > 0 )
            Then ( If IsAKindOf?( B, Base )
                 Then ( B:Color = 3 )
                 Else B:Color = 2 )
            Else If IsAKindOf?( B, Planned )
                 Then ( B:Color = 5 )
                 Else B:Color = 0;
           };
  Bus_Status( );
  } );

    /*************************************
     **** FUNCTION: Commands
     *************************************/
MakeFunction( Commands, [],
  If ( Default:Start #= ON )
   Then ShowImage( RadioButtonGroup4 )
   Else PostMessage( "First, select Data Files, please." ) );

    /*************************************
     **** FUNCTION: Auto
     *************************************/
MakeFunction( Auto, [],
  {
  AppendToList( Commands, Commands, Text5:Title );
  Hide_Auto( );
  } );

    /*************************************
     **** FUNCTION: Simula
     *************************************/
MakeFunction( Simula, [],
  Let [List Commands:Commands]
    {
    HideImage( Text9 );
    HideImage( Text10 );
    OpenWriteFile( Default:Cmd );
    WriteLine( Default:Base );
    WriteLine( Default:Adic );
    For x From 1 To LengthList( List )
        Do WriteLine( GetNthElem( List, x ) );
    WriteLine( "LISTA FLUXO ///" );
    CloseWriteFile( );
    Execute( Default:Prog );
    Commands:Status = TRUE;
    } );

    /*************************************
     **** FUNCTION: Clear_Window
     *************************************/
MakeFunction( Clear_Window, [],
  {
  HideImage( Button8 );
  HideImage( Button2 );
  HideImage( Button1 );
  HideImage( RadioButtonGroup4 );
  HideImage( Drawing1 );
  HideImage( Text1 );
  HideImage( Text2 );
  HideImage( Text3 );
  HideImage( Text9 );
  HideImage( Text10 );
  } );

    /*************************************
     **** FUNCTION: Hide_Adic
     *************************************/
MakeFunction( Hide_Adic, [],
  {
  HideImage( Text4 );
  HideImage( Text5 );
  HideImage( ComboBox2 );
  HideImage( Button4 );
  HideImage( Button5 );
  } );

    /*************************************
     **** FUNCTION: View
     *************************************/
MakeFunction( View, [],
  ShowImage( RadioButtonGroup5 ) );

    /*************************************
     **** FUNCTION: Exclude_List
     *************************************/
MakeFunction( Exclude_List, [],
  {
  Let [List Commands:Simulate]
    Let [N LengthList( Commands:Simulate )]
      For x From 1 To N
          Do Let [Elem GetNthElem( List, x )]
            {
            If Member?( General:Commands, Elem )
             Then RemoveFromList( General:Commands,
Elem );
            If Member?( Synth:Commands, Elem )
             Then RemoveFromList( Synth:Commands,
Elem );
            If Member?( Analy:Commands, Elem )
             Then RemoveFromList( Analy:Commands,
Elem );
            };
  If ( LengthList( General:Commands ) #= 0 )
   Then General:Status = FALSE;
```

```
  ClearList( Commands:Simulate );
  Cancel_Sim( );
  } );

    /************************************
     **** FUNCTION: Show_Auto
     ************************************/
MakeFunction( Show_Auto, [],
  {
  ShowImage( Text6 );
  ShowImage( Slider1 );
  ShowImage( Button3 );
  ShowImage( Button11 );
  } );

    /************************************
     **** FUNCTION: Hide_Auto
     ************************************/
MakeFunction( Hide_Auto, [],
  {
  HideImage( Text5 );
  HideImage( Text6 );
  HideImage( Slider1 );
  HideImage( Button3 );
  HideImage( Button11 );
  } );

    /************************************
     **** FUNCTION: Comandos
     ************************************/
MakeFunction( Comandos, [],
  SendMessage( Commands, Show ) );

    /************************************
     **** FUNCTION: Ret
     ************************************/
MakeFunction( Ret, [],
  {
  Manual:Ret:Ncir = Manual:Ret:Ncir - 1;
  SendMessage( Manual:Ret, MinusCircuit );
  } );

    /************************************
     **** FUNCTION: Cancel_Ret
     ************************************/
MakeFunction( Cancel_Ret, [],
  {
  If Manual:Ret:Overload
   Then {
      Manual:Ret:Color = 4;
      Manual:Ret:Width = 3;
      }
   Else If IsAKindOf?( Manual:Ret, Base )
        Then ( Manual:Ret:Color = 3 )
        Else Manual:Ret:Color = 2;
  Hide_Ret( );
  ShowImage( Drawing1 );
  } );

    /************************************
     **** FUNCTION: Hide_Ret
     ************************************/
MakeFunction( Hide_Ret, [],
  {
  HideImage( Text7 );
  HideImage( Text8 );
  HideImage( ComboBox3 );
  HideImage( Button9 );
  HideImage( Button10 );
  } );

    /************************************
     **** FUNCTION: Cancel_Auto
     ************************************/
MakeFunction( Cancel_Auto, [],
  Hide_Auto( ) );



/*****************************************************/
/**      ALL CLASSES ARE SAVED BELOW          **/
/*****************************************************/

    /************************************
     **** CLASS: Menu
     ************************************/
Menu:X = NULL;
Menu:Y = NULL;
SetSlotOption( Menu:Choices, MULTIPLE );
ClearList( Menu:Choices );
SetSlotOption( Menu:ChoiceNames, MULTIPLE );
ClearList( Menu:ChoiceNames );
SetSlotOption( Menu:Checked, MULTIPLE );
ClearList( Menu:Checked );
SetSlotOption( Menu:Disabled, MULTIPLE );
ClearList( Menu:Disabled );
Menu:DefaultMethod = DisplayCall;

    /************************************
     **** CLASS: FirstLevel
     ************************************/
MakeClass( FirstLevel, Menu );
ClearList( FirstLevel:Choices );
ClearList( FirstLevel:ChoiceNames );
MakeSlot( FirstLevel:Commands );
SetSlotOption( FirstLevel:Commands, MULTIPLE );
```

```
ClearList( FirstLevel:Commands );
SetSlotOption( FirstLevel:Commands, IF_NEEDED, NULL );
SetSlotOption( FirstLevel:Commands, WHEN_ACCESS,
NULL );
SetSlotOption( FirstLevel:Commands, BEFORE_CHANGE,
NULL );
SetSlotOption( FirstLevel:Commands, AFTER_CHANGE,
NULL );
MakeSlot( FirstLevel:Single );
SetSlotOption( FirstLevel:Single, IF_NEEDED, NULL );
SetSlotOption( FirstLevel:Single, WHEN_ACCESS, NULL );
SetSlotOption( FirstLevel:Single, BEFORE_CHANGE, NULL
);
SetSlotOption( FirstLevel:Single, AFTER_CHANGE, NULL );

    /*************************************
     **** CLASS: Image
     *************************************/
Image:X = 100;
SetSlotOption( Image:X, BEFORE_CHANGE,
Update_Image_Struct );
Image:Y = 100;
SetSlotOption( Image:Y, BEFORE_CHANGE,
Update_Image_Struct );
Image:Width = 150;
SetSlotOption( Image:Width, BEFORE_CHANGE,
Update_Image_Struct );
Image:Height = 120;
SetSlotOption( Image:Height, BEFORE_CHANGE,
Update_Image_Struct );
Image:WinType = Make_ImageWindow;
SetSlotOption( Image:Title, IF_NEEDED, NULL );
SetSlotOption( Image:Title, WHEN_ACCESS, NULL );
SetSlotOption( Image:Title, BEFORE_CHANGE,
Update_Image_Struct );
SetSlotOption( Image:Title, AFTER_CHANGE, NULL );
SetSlotOption( Image:RightAction, IF_NEEDED, NULL );
SetSlotOption( Image:RightAction, WHEN_ACCESS, NULL );
SetSlotOption( Image:RightAction, BEFORE_CHANGE,
NULL );
SetSlotOption( Image:RightAction, AFTER_CHANGE, NULL
);
Image:Visible = FALSE;
SetSlotOption( Image:Visible, BEFORE_CHANGE,
Update_Image_Struct );
Image:ShowBorder = TRUE;
SetSlotOption( Image:ShowBorder, BEFORE_CHANGE,
Update_Image_Struct );
Image:Transparent = FALSE;
SetSlotOption( Image:Transparent, BEFORE_CHANGE,
Update_Image_Struct );
SetSlotOption( Image:ForegroundColor, MULTIPLE );
SetValue( Image:ForegroundColor, 0, 0, 0 );
SetSlotOption( Image:ForegroundColor, AFTER_CHANGE,
Update_List_Image_Struct );
SetSlotOption( Image:BackgroundColor, MULTIPLE );
SetValue( Image:BackgroundColor, 255, 255, 255 );
SetSlotOption( Image:BackgroundColor, AFTER_CHANGE,
Update_List_Image_Struct );
Image:Font = "MS Sans Serif";
SetSlotOption( Image:Font, BEFORE_CHANGE,
Update_Image_Struct );
Image:TextSize = 8;
SetSlotOption( Image:TextSize, BEFORE_CHANGE,
Update_Image_Struct );
Image:Underline = FALSE;
SetSlotOption( Image:Underline, BEFORE_CHANGE,
Update_Image_Struct );
Image:Bold = TRUE;
SetSlotOption( Image:Bold, BEFORE_CHANGE,
Update_Image_Struct );
Image:Italic = FALSE;
SetSlotOption( Image:Italic, BEFORE_CHANGE,
Update_Image_Struct );
SetSlotOption( Image:VectorFont, BEFORE_CHANGE,
Update_Image_Struct );
Image:StrikeOut = FALSE;
SetSlotOption( Image:StrikeOut, BEFORE_CHANGE,
Update_Image_Struct );
SetSlotOption( Image:ForegroundColor2, MULTIPLE );
SetValue( Image:ForegroundColor2, 0, 0, 0 );
SetSlotOption( Image:ForegroundColor2, AFTER_CHANGE,
Update_List_Image_Struct );
SetSlotOption( Image:BackgroundColor2, MULTIPLE );
SetValue( Image:BackgroundColor2, 255, 255, 255 );
SetSlotOption( Image:BackgroundColor2, AFTER_CHANGE,
Update_List_Image_Struct );
Image:Font2 = "MS Sans Serif";
SetSlotOption( Image:Font2, BEFORE_CHANGE,
Update_Image_Struct );
Image:TextSize2 = 8;
SetSlotOption( Image:TextSize2, BEFORE_CHANGE,
Update_Image_Struct );
Image:Underline2 = FALSE;
SetSlotOption( Image:Underline2, BEFORE_CHANGE,
Update_Image_Struct );
Image:Bold2 = TRUE;
SetSlotOption( Image:Bold2, BEFORE_CHANGE,
Update_Image_Struct );
Image:Italic2 = FALSE;
SetSlotOption( Image:Italic2, BEFORE_CHANGE,
Update_Image_Struct );
SetSlotOption( Image:TextVector2, BEFORE_CHANGE,
Update_Image_Struct );
Image:StrikeOut2 = FALSE;
```

```
SetSlotOption( Image:StrikeOut2, BEFORE_CHANGE,
Update_Image_Struct );

    /*************************************
     **** CLASS: SlotView
     *************************************/
SetSlotOption( SlotView:Owner, AFTER_CHANGE,
ChangedLink );
SetSlotOption( SlotView:OwnerSlot, AFTER_CHANGE,
ChangedLink );

    /*************************************
     **** CLASS: OutputView
     *************************************/

    /*************************************
     **** CLASS: StateBox
     *************************************/
SetSlotOption( StateBox:AllowableValues, MULTIPLE );
ClearList( StateBox:AllowableValues );
StateBox:Draw = Draw_RectLight;
StateBox:Update = Update_RectLight;

    /*************************************
     **** CLASS: Meter
     *************************************/
Meter:XFormat = FormatValue ( "\%8.2f" );
Meter:Draw = Draw_HMeter;
Meter:Update = Update_HMeter;

    /*************************************
     **** CLASS: InputOutputView
     *************************************/
InputOutputView:TabStop = 0;

    /*************************************
     **** CLASS: ListBox
     *************************************/
ListBox:ShowBorder = FALSE;
ListBox:Sort = TRUE;
ListBox:CurrentTopIndex = 0;
SetSlotOption( ListBox:AllowableValues, MULTIPLE );
ClearList( ListBox:AllowableValues );
ListBox:TabStop = FALSE;

    /*************************************
     **** CLASS: SingleListBox
     *************************************/
SingleListBox:WinType = Make_SingleListBoxWindow;
SingleListBox:Update = Update_SingleLB;

    /*************************************
     **** CLASS: ComboBox
     *************************************/
ComboBox:ShowBorder = FALSE;
ComboBox:Sort = FALSE;
ComboBox:ComboEdit = TRUE;
SetSlotOption( ComboBox:AllowableValues, MULTIPLE );
ClearList( ComboBox:AllowableValues );
ComboBox:Width = 200;
ComboBox:Height = 50;
ComboBox:WinType = Make_ComboBoxWindow;
ComboBox:Update = Update_ComboBox;

    /*************************************
     **** CLASS: MultipleListBox
     *************************************/
SetSlotOption( MultipleListBox:Value, MULTIPLE );
ClearList( MultipleListBox:Value );
MultipleListBox:WinType = Make_MultipleListBoxWindow;
MultipleListBox:Update = Update_MultipleLB;

    /*************************************
     **** CLASS: Edit
     *************************************/
Edit:ShowBorder = TRUE;
Edit:Width = 100;
Edit:Height = 25;
Edit:Multiline = FALSE;
Edit:Password = FALSE;
Edit:WinType = Make_EditWindow;
Edit:Update = Update_Edit;

    /*************************************
     **** CLASS: Slider
     *************************************/
Slider:XFormat = FormatValue ( "\%8.2f" );
Slider:Width = 200;
Slider:Height = 85;
Slider:TabStop = 0;
Slider:WinType = Make_SliderWindow;
Slider:Draw = Draw_Slider;
Slider:Update = Update_Slider;

    /*************************************
     **** CLASS: CheckBox
     *************************************/
CheckBox:ShowBorder = FALSE;
CheckBox:Width = 100;
CheckBox:Height = 25;
CheckBox:WinType = Make_CheckBoxWindow;
CheckBox:Update = Update_CheckBox;

    /*************************************
     **** CLASS: CheckBoxGroup
     *************************************/
```

```
SetSlotOption( CheckBoxGroup:AllowableValues,
MULTIPLE );
ClearList( CheckBoxGroup:AllowableValues );
SetSlotOption( CheckBoxGroup:Value, MULTIPLE );
ClearList( CheckBoxGroup:Value );
CheckBoxGroup:TabStop = FALSE;
CheckBoxGroup:WinType = Make_CBGroupWindow;
CheckBoxGroup:Update = Update_CBGroup;

    /************************************
     **** CLASS: RadioButtonGroup
     ************************************/
SetSlotOption( RadioButtonGroup:AllowableValues,
MULTIPLE );
ClearList( RadioButtonGroup:AllowableValues );
RadioButtonGroup:TabStop = FALSE;
RadioButtonGroup:WinType = Make_RBGroupWindow;
RadioButtonGroup:Update = Update_RBGroup;

    /************************************
     **** CLASS: Button
     ************************************/
SetValue( Button:BackgroundColor, 192, 192, 192 );
Button:ShowBorder = FALSE;
Button:Width = 100;
Button:Height = 25;
Button:TabStop = 0;
Button:WinType = Make_ButtonWindow;
Button:Draw = Draw_Bitmap;

    /************************************
     **** CLASS: Text
     ************************************/
Text:Transparent = FALSE;
Text:Justification = CENTER;
Text:ShowBorder = FALSE;
Text:Width = 100;
Text:Height = 25;
Text:WinType = Make_StaticWindow;

    /************************************
     **** CLASS: Transcript
     ************************************/
Transcript:ProportionalFont = TRUE;
Transcript:VertScroll = AUTO;
Transcript:HorzScroll = AUTO;
Transcript:ShowBorder = TRUE;
Transcript:WinType = Make_TransWindow;

    /************************************
     **** CLASS: LinePlot
     ************************************/
LinePlot:AutoScale = FALSE;
LinePlot:Grid = FALSE;
LinePlot:MinTicks = 3;
LinePlot:MaxTicks = 8;
LinePlot:XFormat = FormatValue ( "\%6.2f" );
LinePlot:YFormat = FormatValue ( "\%6.2f" );
LinePlot:Draw = Draw_LinePlot;
LinePlot:Update = Update_LinePlot;
LinePlot:Print = Print_LinePlot;

    /************************************
     **** CLASS: Bitmap
     ************************************/
Bitmap:FitToScreen = FALSE;
Bitmap:Draw = Draw_Bitmap;

    /************************************
     **** CLASS: Drawing
     ************************************/
Drawing:XLeft = 0;
Drawing:YTop = 0;
Drawing:XRight = 100;
Drawing:YBottom = 100;
Drawing:MetaDraw = FALSE;
Drawing:Draw = Draw_Drawing;
Drawing:Print = Print_Drawing;

    /************************************
     **** CLASS: KWindow
     ************************************/
SetSlotOption( KWindow:BackgroundColor, MULTIPLE );
ClearList( KWindow:BackgroundColor );

    /************************************
     **** CLASS: KSession
     ************************************/
KSession:ReclaimResources = FALSE;
KSession:Visible = FALSE;
KSession:NoMouse = TRUE;
KSession:X = 20;
KSession:Y = 120;
KSession:Width = 465;
KSession:Height = 240;
KSession:Sizebox = TRUE;
KSession:Menu = TRUE;
KSession:Freeze = FALSE;
KSession:Titlebar = TRUE;
KSession:State = HIDDEN;
KSession:WinType = Make_SessionPopupWindow;

    /************************************
     **** CLASS: PowerSystem
     ************************************/
MakeClass( PowerSystem, Root );
```

```
MakeSlot( PowerSystem:x1 );
SetSlotOption( PowerSystem:x1, VALUE_TYPE, NUMBER );
SetSlotOption( PowerSystem:x1, IF_NEEDED, NULL );
SetSlotOption( PowerSystem:x1, WHEN_ACCESS, NULL );
SetSlotOption( PowerSystem:x1, BEFORE_CHANGE, NULL
);
SetSlotOption( PowerSystem:x1, AFTER_CHANGE, NULL );
MakeSlot( PowerSystem:x2 );
SetSlotOption( PowerSystem:x2, VALUE_TYPE, NUMBER );
SetSlotOption( PowerSystem:x2, IF_NEEDED, NULL );
SetSlotOption( PowerSystem:x2, WHEN_ACCESS, NULL );
SetSlotOption( PowerSystem:x2, BEFORE_CHANGE, NULL
);
SetSlotOption( PowerSystem:x2, AFTER_CHANGE, NULL );
MakeSlot( PowerSystem:y1 );
SetSlotOption( PowerSystem:y1, VALUE_TYPE, NUMBER );
SetSlotOption( PowerSystem:y1, IF_NEEDED, NULL );
SetSlotOption( PowerSystem:y1, WHEN_ACCESS, NULL );
SetSlotOption( PowerSystem:y1, BEFORE_CHANGE, NULL
);
SetSlotOption( PowerSystem:y1, AFTER_CHANGE, NULL );
MakeSlot( PowerSystem:y2 );
SetSlotOption( PowerSystem:y2, VALUE_TYPE, NUMBER );
SetSlotOption( PowerSystem:y2, IF_NEEDED, NULL );
SetSlotOption( PowerSystem:y2, WHEN_ACCESS, NULL );
SetSlotOption( PowerSystem:y2, BEFORE_CHANGE, NULL
);
SetSlotOption( PowerSystem:y2, AFTER_CHANGE, NULL );
MakeSlot( PowerSystem:Color );
SetSlotOption( PowerSystem:Color, VALUE_TYPE,
NUMBER );
PowerSystem:Color = 0;
SetSlotOption( PowerSystem:Color, IF_NEEDED, NULL );
SetSlotOption( PowerSystem:Color, WHEN_ACCESS, NULL
);
SetSlotOption( PowerSystem:Color, BEFORE_CHANGE,
NULL );
SetSlotOption( PowerSystem:Color, AFTER_CHANGE,
NULL );
MakeSlot( PowerSystem:Style );
SetSlotOption( PowerSystem:Style, VALUE_TYPE,
NUMBER );
PowerSystem:Style = 1;
SetSlotOption( PowerSystem:Style, IF_NEEDED, NULL );
SetSlotOption( PowerSystem:Style, WHEN_ACCESS, NULL
);
SetSlotOption( PowerSystem:Style, BEFORE_CHANGE,
NULL );
SetSlotOption( PowerSystem:Style, AFTER_CHANGE, NULL
);
MakeSlot( PowerSystem:Width );
SetSlotOption( PowerSystem:Width, VALUE_TYPE,
NUMBER );
PowerSystem:Width = 1;
SetSlotOption( PowerSystem:Width, IF_NEEDED, NULL );
SetSlotOption( PowerSystem:Width, WHEN_ACCESS, NULL
);
SetSlotOption( PowerSystem:Width, BEFORE_CHANGE,
NULL );
SetSlotOption( PowerSystem:Width, AFTER_CHANGE,
NULL );

    /************************************
     **** CLASS: Bus
     ************************************/
MakeClass( Bus, PowerSystem );
MakeSlot( Bus:VoltClass );
SetSlotOption( Bus:VoltClass, VALUE_TYPE, NUMBER );
SetSlotOption( Bus:VoltClass, IF_NEEDED, NULL );
SetSlotOption( Bus:VoltClass, WHEN_ACCESS, NULL );
SetSlotOption( Bus:VoltClass, BEFORE_CHANGE, NULL );
SetSlotOption( Bus:VoltClass, AFTER_CHANGE, NULL );
MakeSlot( Bus:Name );
SetSlotOption( Bus:Name, IF_NEEDED, NULL );
SetSlotOption( Bus:Name, WHEN_ACCESS, NULL );
SetSlotOption( Bus:Name, BEFORE_CHANGE, NULL );
SetSlotOption( Bus:Name, AFTER_CHANGE, NULL );
MakeSlot( Bus:Area );
SetSlotOption( Bus:Area, VALUE_TYPE, NUMBER );
SetSlotOption( Bus:Area, IF_NEEDED, NULL );
SetSlotOption( Bus:Area, WHEN_ACCESS, NULL );
SetSlotOption( Bus:Area, BEFORE_CHANGE, NULL );
SetSlotOption( Bus:Area, AFTER_CHANGE, NULL );
MakeSlot( Bus:Number );
SetSlotOption( Bus:Number, VALUE_TYPE, OBJECT );
SetSlotOption( Bus:Number, ALLOWABLE_CLASSES, Bus
);
SetSlotOption( Bus:Number, IF_NEEDED, NULL );
SetSlotOption( Bus:Number, WHEN_ACCESS, NULL );
SetSlotOption( Bus:Number, BEFORE_CHANGE, NULL );
SetSlotOption( Bus:Number, AFTER_CHANGE, NULL );
MakeSlot( Bus:Connection );
SetSlotOption( Bus:Connection, VALUE_TYPE, BOOLEAN );
Bus:Connection = FALSE;
SetSlotOption( Bus:Connection, IF_NEEDED, NULL );
SetSlotOption( Bus:Connection, WHEN_ACCESS, NULL );
SetSlotOption( Bus:Connection, BEFORE_CHANGE, NULL
);
SetSlotOption( Bus:Connection, AFTER_CHANGE, NULL );
Bus:Color = 5;

    /************************************
     **** CLASS: Generator
     ************************************/
MakeClass( Generator, Bus );
MakeSlot( Generator:Orientation );
```

```
SetSlotOption( Generator:Orientation, IF_NEEDED, NULL );
SetSlotOption( Generator:Orientation, WHEN_ACCESS,
NULL );
SetSlotOption( Generator:Orientation, BEFORE_CHANGE,
NULL );
SetSlotOption( Generator:Orientation, AFTER_CHANGE,
NULL );
MakeSlot( Generator:Year );
SetSlotOption( Generator:Year, VALUE_TYPE, NUMBER );
SetSlotOption( Generator:Year, IF_NEEDED, NULL );
SetSlotOption( Generator:Year, WHEN_ACCESS, NULL );
SetSlotOption( Generator:Year, BEFORE_CHANGE, NULL );
SetSlotOption( Generator:Year, AFTER_CHANGE, NULL );
Generator:Style = 3;
MakeSlot( Generator:Power );
SetSlotOption( Generator:Power, VALUE_TYPE, NUMBER );
SetSlotOption( Generator:Power, IF_NEEDED, NULL );
SetSlotOption( Generator:Power, WHEN_ACCESS, NULL );
SetSlotOption( Generator:Power, BEFORE_CHANGE, NULL
);
SetSlotOption( Generator:Power, AFTER_CHANGE, NULL );

    /*************************************
     **** CLASS: Load
     *************************************/
MakeClass( Load, Bus );
MakeSlot( Load:Orientation );
SetSlotOption( Load:Orientation, IF_NEEDED, NULL );
SetSlotOption( Load:Orientation, WHEN_ACCESS, NULL );
SetSlotOption( Load:Orientation, BEFORE_CHANGE, NULL
);
SetSlotOption( Load:Orientation, AFTER_CHANGE, NULL );
MakeSlot( Load:Year );
SetSlotOption( Load:Year, VALUE_TYPE, NUMBER );
SetSlotOption( Load:Year, IF_NEEDED, NULL );
SetSlotOption( Load:Year, WHEN_ACCESS, NULL );
SetSlotOption( Load:Year, BEFORE_CHANGE, NULL );
SetSlotOption( Load:Year, AFTER_CHANGE, NULL );
Load:Style = 3;
MakeSlot( Load:Power );
SetSlotOption( Load:Power, VALUE_TYPE, NUMBER );
SetSlotOption( Load:Power, IF_NEEDED, NULL );
SetSlotOption( Load:Power, WHEN_ACCESS, NULL );
SetSlotOption( Load:Power, BEFORE_CHANGE, NULL );
SetSlotOption( Load:Power, AFTER_CHANGE, NULL );

    /*************************************
     **** CLASS: Network
     *************************************/
MakeClass( Network, PowerSystem );

/************** METHOD: PlusCircuit  **************/
MakeMethod( Network, PlusCircuit, [],
  {
  If ( Self:Ncir > Self:Circuits )
   Then {
      PostMessage( "Faixa de passagem esgotada." );
      Self:Ncir = Self:Circuits;
      }
   Else {
      Manual:Adic:Style = 1;
      SendMessage( Self, Connection, InitBus,
Manual:Adic:InitBus );
      SendMessage( Self, Connection, EndBus,
Manual:Adic:EndBus );
      AppendToList( Commands, Commands, Text5:Title );
      AppendToList( Self, Years, Text3:Title );
      ShowImage( Drawing1 );
      };
  Hide_Adic( );
  } );

/************** METHOD: MinusCircuit  **************/
MakeMethod( Network, MinusCircuit, [],
  {
  If ( Self:Ncir < 0 )
   Then PostMessage( "Não há Circuitos a Excluir." )
   Else {
      If ( Self:Ncir #= 0 )
       Then ( If IsAKindOf?( Manual:Ret, Base )
            Then ( Manual:Ret:Color = 0 )
            Else {
               Manual:Ret:Color = 5;
               Manual:Ret:Style = 2;
               SendMessage( Self, Desconnection, InitBus,
                  Manual:Ret:InitBus );
               SendMessage( Self, Desconnection,
EndBus,
                  Manual:Ret:EndBus );
               } )
       Else {
          If IsAKindOf?( Manual:Ret, Base )
           Then ( Manual:Ret:Color = 3 )
           Else Manual:Ret:Color = 2;
          };
      AppendToList( Commands, Commands, Text8:Title );
      RemoveFromList( Self:Years, Text3:Title );
      ShowImage( Drawing1 );
      };
  Hide_Ret( );
  } );

/************** METHOD: Limit  **************/
MakeMethod( Network, Limit, [slot new_value ],
  {
  If ( new_value > Self:FlowLimit )
```

```
    Then {
        SetValue( Self:Overload, TRUE );
        Study:Overload = Study:Overload + new_value -
Self:FlowLimit;
        }
    Else SetValue( Self:Overload, FALSE );
  new_value;
  } );

/************** METHOD: Connection  **************/
MakeMethod( Network, Connection, [slot new_bus ],
  {
  new_bus:Connection = TRUE;
  new_bus:Color = 0;
  new_bus;
  } );

/************** METHOD: Desconnection  **************/
MakeMethod( Network, Desconnection, [slot new_bus ],
  {
  new_bus:Connection = FALSE;
  new_bus:Color = 5;
  new_bus;
  } );
MakeSlot( Network:Circuits );
SetSlotOption( Network:Circuits, VALUE_TYPE, NUMBER );
SetSlotOption( Network:Circuits, IF_NEEDED, NULL );
SetSlotOption( Network:Circuits, WHEN_ACCESS, NULL );
SetSlotOption( Network:Circuits, BEFORE_CHANGE, NULL
);
SetSlotOption( Network:Circuits, AFTER_CHANGE, NULL );
MakeSlot( Network:EndBus );
SetSlotOption( Network:EndBus, VALUE_TYPE, OBJECT );
SetSlotOption( Network:EndBus, ALLOWABLE_CLASSES,
Bus );
SetSlotOption( Network:EndBus, IF_NEEDED, NULL );
SetSlotOption( Network:EndBus, WHEN_ACCESS, NULL );
SetSlotOption( Network:EndBus, BEFORE_CHANGE, NULL
);
SetSlotOption( Network:EndBus, AFTER_CHANGE, NULL );
MakeSlot( Network:FlowLimit );
SetSlotOption( Network:FlowLimit, VALUE_TYPE, NUMBER
);
SetSlotOption( Network:FlowLimit, IF_NEEDED, NULL );
SetSlotOption( Network:FlowLimit, WHEN_ACCESS, NULL
);
SetSlotOption( Network:FlowLimit, BEFORE_CHANGE,
NULL );
SetSlotOption( Network:FlowLimit, AFTER_CHANGE, NULL
);
MakeSlot( Network:InitBus );
SetSlotOption( Network:InitBus, VALUE_TYPE, OBJECT );
SetSlotOption( Network:InitBus, ALLOWABLE_CLASSES,
Bus );
SetSlotOption( Network:InitBus, IF_NEEDED, NULL );
SetSlotOption( Network:InitBus, WHEN_ACCESS, NULL );
SetSlotOption( Network:InitBus, BEFORE_CHANGE, NULL );
SetSlotOption( Network:InitBus, AFTER_CHANGE, NULL );
MakeSlot( Network:x3 );
SetSlotOption( Network:x3, VALUE_TYPE, NUMBER );
SetSlotOption( Network:x3, IF_NEEDED, NULL );
SetSlotOption( Network:x3, WHEN_ACCESS, NULL );
SetSlotOption( Network:x3, BEFORE_CHANGE, NULL );
SetSlotOption( Network:x3, AFTER_CHANGE, NULL );
MakeSlot( Network:x4 );
SetSlotOption( Network:x4, VALUE_TYPE, NUMBER );
SetSlotOption( Network:x4, IF_NEEDED, NULL );
SetSlotOption( Network:x4, WHEN_ACCESS, NULL );
SetSlotOption( Network:x4, BEFORE_CHANGE, NULL );
SetSlotOption( Network:x4, AFTER_CHANGE, NULL );
MakeSlot( Network:y3 );
SetSlotOption( Network:y3, VALUE_TYPE, NUMBER );
SetSlotOption( Network:y3, IF_NEEDED, NULL );
SetSlotOption( Network:y3, WHEN_ACCESS, NULL );
SetSlotOption( Network:y3, BEFORE_CHANGE, NULL );
SetSlotOption( Network:y3, AFTER_CHANGE, NULL );
MakeSlot( Network:y4 );
SetSlotOption( Network:y4, VALUE_TYPE, NUMBER );
SetSlotOption( Network:y4, IF_NEEDED, NULL );
SetSlotOption( Network:y4, WHEN_ACCESS, NULL );
SetSlotOption( Network:y4, BEFORE_CHANGE, NULL );
SetSlotOption( Network:y4, AFTER_CHANGE, NULL );
MakeSlot( Network:Ncir );
SetSlotOption( Network:Ncir, VALUE_TYPE, NUMBER );
Network:Ncir = 0;
SetSlotOption( Network:Ncir, IF_NEEDED, NULL );
SetSlotOption( Network:Ncir, WHEN_ACCESS, NULL );
SetSlotOption( Network:Ncir, BEFORE_CHANGE, NULL );
SetSlotOption( Network:Ncir, AFTER_CHANGE, NULL );
MakeSlot( Network:Flow );
SetSlotOption( Network:Flow, VALUE_TYPE, NUMBER );
SetSlotOption( Network:Flow, IF_NEEDED, NULL );
SetSlotOption( Network:Flow, WHEN_ACCESS, NULL );
SetSlotOption( Network:Flow, BEFORE_CHANGE, Limit );
SetSlotOption( Network:Flow, AFTER_CHANGE, NULL );
MakeSlot( Network:Overload );
SetSlotOption( Network:Overload, VALUE_TYPE, BOOLEAN
);
Network:Overload = FALSE;
SetSlotOption( Network:Overload, IF_NEEDED, NULL );
SetSlotOption( Network:Overload, WHEN_ACCESS, NULL );
SetSlotOption( Network:Overload, BEFORE_CHANGE,
NULL );
SetSlotOption( Network:Overload, AFTER_CHANGE, NULL
);
```

```
MakeSlot( Network:Years );
SetSlotOption( Network:Years, MULTIPLE );
ClearList( Network:Years );
SetSlotOption( Network:Years, IF_NEEDED, NULL );
SetSlotOption( Network:Years, WHEN_ACCESS, NULL );
SetSlotOption( Network:Years, BEFORE_CHANGE, NULL );
SetSlotOption( Network:Years, AFTER_CHANGE, NULL );

    /************************************
     **** CLASS: Base
     ************************************/
MakeClass( Base, Network );

/************ METHOD: Connection  ************/
MakeMethod( Base, Connection, [slot new_bus ],
  {
  SetValue( new_bus:Connection, TRUE );
  SetValue( new_bus:Color, 0 );
  new_bus;
  } );
SetSlotOption( Base:EndBus, BEFORE_CHANGE,
Connection );
SetSlotOption( Base:InitBus, BEFORE_CHANGE,
Connection );

    /************************************
     **** CLASS: Line
     ************************************/
MakeClass( Line, Base );
MakeSlot( Line:Broken );
SetSlotOption( Line:Broken, VALUE_TYPE, BOOLEAN );
Line:Broken = FALSE;
SetSlotOption( Line:Broken, IF_NEEDED, NULL );
SetSlotOption( Line:Broken, WHEN_ACCESS, NULL );
SetSlotOption( Line:Broken, BEFORE_CHANGE, NULL );
SetSlotOption( Line:Broken, AFTER_CHANGE, NULL );

    /************************************
     **** CLASS: Transformer
     ************************************/
MakeClass( Transformer, Base );

    /************************************
     **** CLASS: Planned
     ************************************/
MakeClass( Planned, Network );
Planned:Style = 2;
Planned:Color = 5;

    /************************************
     **** CLASS: LineP
     ************************************/
MakeClass( LineP, Planned );
MakeSlot( LineP:Broken );
SetSlotOption( LineP:Broken, VALUE_TYPE, BOOLEAN );
LineP:Broken = FALSE;
SetSlotOption( LineP:Broken, IF_NEEDED, NULL );
SetSlotOption( LineP:Broken, WHEN_ACCESS, NULL );
SetSlotOption( LineP:Broken, BEFORE_CHANGE, NULL );
SetSlotOption( LineP:Broken, AFTER_CHANGE, NULL );

    /************************************
     **** CLASS: TransformerP
     ************************************/
MakeClass( TransformerP, Planned );

    /************************************
     **** CLASS: Commands
     ************************************/
MakeClass( Commands, Root );

/************ METHOD: Check  ************/
MakeMethod( Commands, Check, [slot Single ],
  {
  If ( Single #= SÍNTESE )
   Then ShowImage( RadioButtonGroup6 );
  If ( Single #= ANÁLISE )
   Then ShowImage( RadioButtonGroup2 );
  If ( Single #= GERAL )
   Then SendMessage( General, Show );
  If ( Single #= CANCELA )
   Then HideImage( RadioButtonGroup4 );
  Single;
  } );

/************ METHOD: Update  ************/
MakeMethod( Commands, Update, [],
  If KnownValue?( Self:Commands )
   Then {
      OpenWriteFile( Default:Cmd, APPEND );
      WriteLine( GetNthElem( Self:Commands, LengthList(
Self:Commands ) ) );
      CloseWriteFile( );
      } );

/************ METHOD: Show  ************/
MakeMethod( Commands, Show, [],
  {
  If General:Status
   Then {
      SetValue( RadioButtonGroup4:AllowableValues,
SÍNTESE,
        ANÁLISE, GERAL, CANCELA );
      ShowImage( Button2 );
      ShowImage( Button1 );
      }
```

```
    Else {
      SetValue( RadioButtonGroup4:AllowableValues,
GERAL,
        CANCELA );
      HideImage( Button2 );
      HideImage( Button1 );
      };
  ShowImage( RadioButtonGroup4 );
  } );
MakeSlot( Commands:Single );
Commands:Single = CANCELA;
SetSlotOption( Commands:Single, IF_NEEDED, NULL );
SetSlotOption( Commands:Single, WHEN_ACCESS, NULL );
SetSlotOption( Commands:Single, BEFORE_CHANGE,
Check );
SetSlotOption( Commands:Single, AFTER_CHANGE, NULL
);
SetSlotOption( Commands:Single, IMAGE,
RadioButtonGroup4 );
MakeSlot( Commands:Commands );
SetSlotOption( Commands:Commands, MULTIPLE );
ClearList( Commands:Commands );
SetSlotOption( Commands:Commands, IF_NEEDED, NULL
);
SetSlotOption( Commands:Commands, WHEN_ACCESS,
NULL );
SetSlotOption( Commands:Commands, BEFORE_CHANGE,
NULL );
SetSlotOption( Commands:Commands, AFTER_CHANGE,
Update );
SetSlotOption( Commands:Commands, IMAGE,
MultipleListBox1 );
MakeSlot( Commands:Status );
SetSlotOption( Commands:Status, VALUE_TYPE,
BOOLEAN );
Commands:Status = TRUE;
SetSlotOption( Commands:Status, IF_NEEDED, NULL );
SetSlotOption( Commands:Status, WHEN_ACCESS, NULL );
SetSlotOption( Commands:Status, BEFORE_CHANGE,
NULL );
SetSlotOption( Commands:Status, AFTER_CHANGE, NULL
);

    /*************************************
     **** CLASS: General
     *************************************/
MakeClass( General, Commands );

/************** METHOD: Reset  **************/
/* "Reinicializa a presente seção de estudo." */
MakeMethod( General, Reset, [],
  {
  PostMessage( GetMethodComment( General, Reset ) );
  Reset( );
  ForAll [ B|Base ]
      {
      B:Color = 0;
      B:Width = 1;
      };
  ForAll [ P|Planned ]
      {
      P:Color = 5;
      P:Width = 1;
      };
  Bus_Status( );
  HideImage( Text9 );
  HideImage( Text10 );
  HideImage( RadioButtonGroup1 );
  HideImage( ComboBox1 );
  SendMessage( Commands, Show );
  } );
SetMethodComment(General, Reset, "Reinicializa a
presente seção de estudo." );

/************** METHOD: Check  **************/
MakeMethod( General, Check, [slot Single ],
  {
  If ( Single #= "NOVO ANO" )
   Then ShowImage( ComboBox1 );
  If ( Single #= "REINICIALIZA ESTUDO" )
   Then SendMessage( Self, Reset );
  If ( Single #= "SIMULA PLANO" )
   Then Simula( );
  If ( Single #= CANCELA )
   Then {
     HideImage( RadioButtonGroup1 );
     HideImage( ComboBox1 );
     };
  Single;
  } );

/************** METHOD: Show  **************/
MakeMethod( General, Show, [],
  {
  If General:Status
   Then SetValue( RadioButtonGroup1:AllowableValues,
"NOVO ANO",
        "REINICIALIZA ESTUDO", "SIMULA PLANO",
CANCELA )
   Else SetValue( RadioButtonGroup1:AllowableValues,
"NOVO ANO",
        CANCELA );
  ShowImage( RadioButtonGroup1 );
  } );
MakeSlot( General:Status );
SetSlotOption( General:Status, INHERIT, FALSE );
```

```
SetSlotOption( General:Status, VALUE_TYPE, BOOLEAN );
General:Status = FALSE;
SetSlotOption( General:Status, IF_NEEDED, NULL );
SetSlotOption( General:Status, WHEN_ACCESS, NULL );
SetSlotOption( General:Status, BEFORE_CHANGE, NULL );
SetSlotOption( General:Status, AFTER_CHANGE, NULL );
ClearList( General:Commands );
General:Single = CANCELA;
SetSlotOption( General:Single, IMAGE, RadioButtonGroup1
);

    /*************************************
     **** CLASS: Analy
     *************************************/
MakeClass( Analy, Commands );

/************** METHOD: Check  **************/
MakeMethod( Analy, Check, [slot Single ],
  {
  If ( Single #= "LISTA ADIC" )
   Then SendMessage( Analysis, ListaAdic );
  If ( Single #= "LISTA AECON" )
   Then SendMessage( Analysis, ListaAecon );
  If ( Single #= "LISTA BARRAS" )
   Then SendMessage( Analysis, ListaBarras );
  If ( Single #= "LISTA CIRCEX" )
   Then SendMessage( Analysis, ListaCircex );
  If ( Single #= "LISTA CIRCOB" )
   Then SendMessage( Analysis, ListaCircob );
  If ( Single #= "LISTA CIRCPL" )
   Then SendMessage( Analysis, ListaCircpl );
  If ( Single #= "LISTA CIRMAX" )
   Then SendMessage( Analysis, ListaCirmax );
  If ( Single #= "LISTA CONTIN" )
   Then SendMessage( Analysis, ListaContin );
  If ( Single #= "LISTA EXPAN" )
   Then SendMessage( Analysis, ListaExpan );
  If ( Single #= "LISTA FLADIC" )
   Then SendMessage( Analysis, ListaFladic );
  If ( Single #= "LISTA FLUXO" )
   Then SendMessage( Analysis, ListaFluxo );
  If ( Single #= "LISTA RETIRA" )
   Then SendMessage( Analysis, ListaRetira );
  If ( Single #= CANCELA )
   Then HideImage( RadioButtonGroup2 );
  Single;
  } );
ClearList( Analy:Commands );
Analy:Single = CANCELA;
SetSlotOption( Analy:Single, IMAGE, RadioButtonGroup2 );

    /*************************************
     **** CLASS: Synth
     *************************************/
MakeClass( Synth, Commands );

/************** METHOD: Check  **************/
MakeMethod( Synth, Check, [slot Single ],
  {
  If ( Single #= MANUAL )
   Then ShowImage( RadioButtonGroup7 );
  If ( Single #= AUTOMÁTICO )
   Then ShowImage( RadioButtonGroup8 );
  If ( Single #= CANCELA )
   Then HideImage( RadioButtonGroup6 );
  Single;
  } );
ClearList( Synth:Commands );
Synth:Single = CANCELA;
SetSlotOption( Synth:Single, IMAGE, RadioButtonGroup6 );



/*******************************************************/
/**       ALL INSTANCES ARE SAVED BELOW          **/
/*******************************************************/
MakeSlot( Global:Check_Flow );
SetSlotOption( Global:Check_Flow, INHERIT, FALSE );
SetSlotOption( Global:Check_Flow, MULTIPLE );
SetValue( Global:Check_Flow, Overload_Flow,
Normal_Flow, Emergency_Normal, Emergency_Over );
SetSlotOption( Global:Check_Flow, IF_NEEDED, NULL );
SetSlotOption( Global:Check_Flow, WHEN_ACCESS, NULL
);
SetSlotOption( Global:Check_Flow, BEFORE_CHANGE,
NULL );
SetSlotOption( Global:Check_Flow, AFTER_CHANGE, NULL
);

    /*************************************
     **** INSTANCE: SESSION
     *************************************/
SESSION:X = 101;
SESSION:Y = 47;
SESSION:Title = SESSION;
SESSION:SessionNumber = 0;
SESSION:Width = 575;
SESSION:Height = 488;
SESSION:Visible = FALSE;
SESSION:State = HIDDEN;
SetValue( SESSION:BackgroundColor, 0, 0, 255 );
SESSION:Menu = TRUE;
SESSION:Titlebar = TRUE;
SESSION:Sizebox = TRUE;
SESSION:Freeze = FALSE;
SESSION:Action = Work;
```

```
ResetWindow ( SESSION );

    /************************************
     **** INSTANCE: Options
     ************************************/
MakeInstance( Options, Menu );

/************** METHOD: Analysis  **************/
MakeMethod( Options, Analysis, [],
  PostMessage( "Comandos de Análise" ) );

/************** METHOD: Synthesis  **************/
MakeMethod( Options, Synthesis, [],
  PostMessage( "Comandos de Síntese" ) );

/************** METHOD: View  **************/
MakeMethod( Options, View, [],
  PostMessage( "Display file contents." ) );
Options:X = 88;
Options:Y = 114;
SetValue( Options:Choices, Synthesis, Analysis, View );
SetValue( Options:ChoiceNames, S&ynthesis, &Analysis,
&View );

    /************************************
     **** INSTANCE: Synthesis
     ************************************/
MakeInstance( Synthesis, FirstLevel );

/************** METHOD: Adic  **************/
/* "ADIC - Add circuits foreseen in the expansion planning."
*/
MakeMethod( Synthesis, Adic, [circuit ],
  If ( General:Status #= FALSE )
    Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
    Else ShowImage( ComboBox2 ) );
SetMethodComment(Synthesis, Adic, "ADIC - Add circuits
foreseen in the expansion planning." );

/************** METHOD: Remove  **************/
/* "REMOVE - Remove circuits that were add or
incorporated." */
MakeMethod( Synthesis, Remove, [],
  If ( General:Status #= FALSE )
    Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
    Else {
      PostInputForm( [ CANCELA ], GetMethodComment(
Synthesis,
                    Remove ), Synthesis,
        Single, "Choose circuits to remove: " );
      If Not( Null?( Synthesis:Single ) )
        Then {
          AppendToList( Synthesis:Commands,
            "REMOVE " # Synthesis:Single );
          SetValue( Synthesis:Single, NULL );
          };
      } );
SetMethodComment(Synthesis, Remove, "REMOVE -
Remove circuits that were add or incorporated." );

/************** METHOD: Retira  **************/
/* "RETIRA - Retract circuits that were add." */
MakeMethod( Synthesis, Retira, [],
  If ( General:Status #= FALSE )
    Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
    Else {
      PostInputForm( [ CANCELA ], GetMethodComment(
Synthesis,
                    Retira ), Synthesis,
        Single, "Choose circuits to retract: " );
      If Not( Null?( Synthesis:Single ) )
        Then {
          AppendToList( Synthesis:Commands,
            "RETIRA " # Synthesis:Single );
          SetValue( Synthesis:Single, NULL );
          };
      } );
SetMethodComment(Synthesis, Retira, "RETIRA - Retract
circuits that were add." );

/************** METHOD: Incorp  **************/
/* "INCORP - Incorporate circuits to the study network." */
MakeMethod( Synthesis, Incorp, [],
  If ( General:Status #= FALSE )
    Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
    Else {
      PostInputForm( [ CANCELA ], GetMethodComment(
Synthesis,
                    Incorp ), Synthesis:Single,
        "Choose circuits to incorporate: " );
      If Not( Null?( Synthesis:Single ) )
        Then {
          AppendToList( Synthesis:Commands,
            "INCORP " # Synthesis:Single );
          SetValue( Synthesis:Single, NULL );
          };
      } );
SetMethodComment(Synthesis, Incorp, "INCORP -
Incorporate circuits to the study network." );

/************** METHOD: Reforc  **************/
```

```
/* "REFORC - Reinforce automatically the system for single
contingencies." */
MakeMethod( Synthesis, Reforc, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Synthesis,
                    Reforc ), Synthesis,
       Single, "Choose circuits to reinforce: " );
     If Not( Null?( Synthesis:Single ) )
      Then {
        AppendToList( Synthesis:Commands,
          "REFORC " # Synthesis:Single );
        SetValue( Synthesis:Single, NULL );
        };
     } );
SetMethodComment(Synthesis, Reforc, "REFORC -
Reinforce automatically the system for single
contingencies." );

/************** METHOD: Antref  **************/
/* "ANTREF - Promote automatic anticipation of future
reinforcement for single contingencies." */
MakeMethod( Synthesis, Antref, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Synthesis,
                    Antref ), Synthesis,
       Single, "Choose reinforcement to anticipate: " );
     If Not( Null?( Synthesis:Single ) )
      Then {
        AppendToList( Synthesis:Commands,
          "ANTREF " # Synthesis:Single );
        SetValue( Synthesis:Single, NULL );
        };
     } );
SetMethodComment(Synthesis, Antref, "ANTREF - Promote
automatic anticipation of future reinforcement for single
contingencies." );

/************** METHOD: Antadi  **************/
/* "ANTADI - Anticipate circuits incorporated in future
years." */
MakeMethod( Synthesis, Antadi, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Synthesis,
                    Antadi ), Synthesis,
       Single, "Choose circuits to anticipate: " );
     If Not( Null?( Synthesis:Single ) )
      Then {
        AppendToList( Synthesis:Commands,
          "ANTADI " # Synthesis:Single );
        SetValue( Synthesis:Single, NULL );
        };
     } );
SetMethodComment(Synthesis, Antadi, "ANTADI -
Anticipate circuits incorporated in future years." );

/************** METHOD: Retcon  **************/
/* "RETCON - Remove automatically the redundant
reinforcements after contingency's analysis." */
MakeMethod( Synthesis, Retcon, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Synthesis,
                    Retcon ), Synthesis,
       Single, "Choose reinforcements to remove: " );
     If Not( Null?( Synthesis:Single ) )
      Then {
        AppendToList( Synthesis:Commands,
          "RETCON " # Synthesis:Single );
        SetValue( Synthesis:Single, NULL );
        };
     } );
SetMethodComment(Synthesis, Retcon, "RETCON -
Remove automatically the redundant reinforcements after
contingency's analysis." );

/************** METHOD: Update  **************/
MakeMethod( Synthesis, Update, [],
  If KnownValue?( Synthesis:Commands )
   Then {
     OpenWriteFile( Default:Cmd, APPEND );
     WriteLine( GetNthElem( Synthesis:Commands,
           LengthList( Synthesis:Commands ) ) );
     CloseWriteFile( );
     Execute( Default:Prog );
     } );
SetValue( Synthesis:ChoiceNames, &Adic, &Incorp,
A&ntadi, -, &Retira, R&emove, -, Re&forc, An&tref, Re&tcon
);
SetValue( Synthesis:Commands, "ADIC 405/419", "ADIC
431/411", "ADIC 425/422", "ADIC 425/402", "INCORP
CANCEL", "ANTADI CANCEL", "RETIRA 432/416" );
```

```
SetSlotOption( Synthesis:Commands, BEFORE_CHANGE,
NULL );
SetSlotOption( Synthesis:Commands, AFTER_CHANGE,
Update );
Synthesis:Single = NULL;
SetSlotOption( Synthesis:Single, WHEN_ACCESS, NULL );
SetSlotOption( Synthesis:Single, BEFORE_CHANGE, NULL
);
SetSlotOption( Synthesis:Single, AFTER_CHANGE, NULL );
SetSlotOption( Synthesis:Single, IMAGE,
RadioButtonGroup3 );
SetValue( Synthesis:Choices, Adic, Incorp, Antadi, -, Retira,
Remove, -, Reforc, Antref, Retcon );

    /************************************
     **** INSTANCE: Analysis
     ************************************/
MakeInstance( Analysis, FirstLevel );

/************** METHOD: ListaAdic  **************/
/* "LISTA ADIC - List most promising additions." */
MakeMethod( Analysis, ListaAdic, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
      PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                        ListaAdic ), Analysis,
        Single, "Define range of addition circuits to list: " );
      If Not( Null?( Analysis:Single ) )
       Then {
         AppendToList( Analy:Commands, "LISTA ADIC "
#
                              Analysis:Single );
         SetValue( Analysis:Single, NULL );
         };
      } );
SetMethodComment(Analysis, ListaAdic, "LISTA ADIC - List
most promising additions." );

/************** METHOD: ListaAecon  **************/
/* "LISTA AECON - List economic analysis for a certain
year." */
MakeMethod( Analysis, ListaAecon, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
      PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                        ListaAecon ), Analysis,
        Single, "Define year for economic analysis: " );
      If Not( Null?( Analysis:Single ) )
       Then {
         AppendToList( Analy:Commands, "LISTA AECON
" #
                              Analysis:Single );
         SetValue( Analysis:Single, NULL );
         };
      } );
SetMethodComment(Analysis, ListaAecon, "LISTA AECON -
List economic analysis for a certain year." );

/************** METHOD: ListaBarras  **************/
/* "LISTA BARRAS - List bar data by type, area and volt
level." */
MakeMethod( Analysis, ListaBarras, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
      PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                        ListaBarras ), Analysis,
        Single, "Define range of bars to list: " );
      If Not( Null?( Analysis:Single ) )
       Then {
         AppendToList( Analy:Commands, "LISTA
BARRAS "
                              # Analysis:Single );
         SetValue( Analysis:Single, NULL );
         };
      } );
SetMethodComment(Analysis, ListaBarras, "LISTA
BARRAS - List bar data by type, area and volt level." );

/************** METHOD: ListaCircex  **************/
/* "LISTA CIRCEX - List data of existent circuits." */
MakeMethod( Analysis, ListaCircex, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
      PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                        ListaCircex ), Analysis,
        Single, "Define range of existent circuits to list: " );
      If Not( Null?( Analysis:Single ) )
       Then {
         AppendToList( Analy:Commands, "LISTA
CIRCEX" #
                              Analysis:Single );
         SetValue( Analysis:Single, NULL );
         };
      } );
```

```
SetMethodComment(Analysis, ListaCircex, "LISTA CIRCEX
- List data of existent circuits." );

/************** METHOD: ListaCircob  **************/
/* "LISTA CIRCOB - List data of obligatory circuits for the
current year." */
MakeMethod( Analysis, ListaCircob, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                    ListaCircob ), Analysis,
       Single, "Define range of obligatory circuits to list: "
);
     If Not( Null?( Analysis:Single ) )
      Then {
        AppendToList( Analy:Commands, "LISTA
CIRCOB "
                           # Analysis:Single );
        SetValue( Analysis:Single, NULL );
        };
     } );
SetMethodComment(Analysis, ListaCircob, "LISTA CIRCOB
- List data of obligatory circuits for the current year." );

/************** METHOD: ListaCircpl  **************/
/* "LISTA CIRCPL - List planned circuits." */
MakeMethod( Analysis, ListaCircpl, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                    ListaCircpl ), Analysis,
       Single, "Define planned circuits to list: " );
     If Not( Null?( Analysis:Single ) )
      Then {
        AppendToList( Analy:Commands, "LISTA
CIRCPL "
                           # Analysis:Single );
        SetValue( Analysis:Single, NULL );
        };
     } );
SetMethodComment(Analysis, ListaCircpl, "LISTA CIRCPL -
List planned circuits." );

/************** METHOD: ListaCirmax  **************/
/* "LISTA CIRMAX - List circuits that don't support
duplication." */
MakeMethod( Analysis, ListaCirmax, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                    ListaCirmax ), Analysis,
       Single, "Define range of circuits to list: " );
     If Not( Null?( Analysis:Single ) )
      Then {
        AppendToList( Analy:Commands, "LISTA
CIRMAX "
                           # Analysis:Single );
        SetValue( Analysis:Single, NULL );
        };
     } );
SetMethodComment(Analysis, ListaCirmax, "LISTA
CIRMAX - List circuits that don't support duplication." );

/************** METHOD: ListaDiag  **************/
/* "LISTA DIAG - List diagnosis of power system's
changes." */
MakeMethod( Analysis, ListaDiag, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostMessage( GetMethodComment( Analysis,
ListaDiag ) );
     AppendToList( Analy:Commands, "LISTA DIAG" );
     } );
SetMethodComment(Analysis, ListaDiag, "LISTA DIAG - List
diagnosis of power system's changes." );

/************** METHOD: ListaContin  **************/
/* "LISTA CONTIN - List more severe contingencies for the
circuits specified." */
MakeMethod( Analysis, ListaContin, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                    ListaContin ), Analysis,
       Single, "Define range of circuits to list: " );
     If Not( Null?( Analysis:Single ) )
      Then {
        AppendToList( Analy:Commands, "LISTA
CONTIN "
                           # Analysis:Single );
        SetValue( Analysis:Single, NULL );
        };
```

```
    } );
SetMethodComment(Analysis, ListaContin, "LISTA CONTIN
- List more severe contingencies for the circuits specified."
);

/************** METHOD: ListaExpan  **************/
/* "LISTA EXPAN - List the actual synthesis' state for a
certain year." */
MakeMethod( Analysis, ListaExpan, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                    ListaExpan ), Analysis,
       Single, "Define year of study: " );
     If Not( Null?( Analysis:Single ) )
      Then {
        AppendToList( Analy:Commands, "LISTA EXPAN
" #
                                Analysis:Single );
        SetValue( Analysis:Single, NULL );
        };
     } );
SetMethodComment(Analysis, ListaExpan, "LISTA EXPAN -
List the actual synthesis' state for a certain year." );

/************** METHOD: ListaFladic  **************/
/* "LISTA FLADIC - List power flow data in the added
circuits." */
MakeMethod( Analysis, ListaFladic, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                    ListaFladic ), Analysis,
       Single, "Define range of circuits to list: " );
     If Not( Null?( Analysis:Single ) )
      Then {
        AppendToList( Analy:Commands, "LISTA
FLADIC "
                                # Analysis:Single );
        SetValue( Analysis:Single, NULL );
        };
     } );
SetMethodComment(Analysis, ListaFladic, "LISTA FLADIC -
List power flow data in the added circuits." );

/************** METHOD: ListaFluxo  **************/
/* "LISTA FLUXO - List power flow data in the specified
circuits." */
MakeMethod( Analysis, ListaFluxo, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                    ListaFluxo ), Analysis,
       Single, "Define range of circuits to list: " );
     If Not( Null?( Analysis:Single ) )
      Then {
        AppendToList( Analy:Commands, "LISTA FLUXO
" #
                                Analysis:Single );
        SetValue( Analysis:Single, NULL );
        };
     } );
SetMethodComment(Analysis, ListaFluxo, "LISTA FLUXO -
List power flow data in the specified circuits." );

/************** METHOD: ListaRetira  **************/
/* "LISTA RETIRA - List the best circuits' candidates to been
retract from the power system." */
MakeMethod( Analysis, ListaRetira, [],
  If ( General:Status #= FALSE )
   Then PostMessage( "Primeiro escolha o ano de
planejamento, por favor." )
   Else {
     PostInputForm( [ CANCELA ], GetMethodComment(
Analysis,
                    ListaRetira ), Analysis,
       Single, "Define range of circuits to list: " );
     If Not( Null?( Analysis:Single ) )
      Then {
        AppendToList( Analy:Commands, "LISTA
RETIRA "
                                # Analysis:Single );
        SetValue( Analysis:Single, NULL );
        };
     } );
SetMethodComment(Analysis, ListaRetira, "LISTA RETIRA -
List the best circuits' candidates to been retract from the
power system." );

/************** METHOD: Update  **************/
MakeMethod( Analysis, Update, [],
  If KnownValue?( Analysis:Commands )
   Then {
     OpenWriteFile( Default:Cmd, APPEND );
     WriteLine( GetNthElem( Analysis:Commands,
            LengthList( Analysis:Commands ) ) );
```

```
      CloseWriteFile( );
      Execute( Default:Prog );
      } );
SetValue( Analysis:Choices, ListaAdic, ListaAecon,
ListaBarras, ListaCircex, ListaCircob, ListaCircpl,
ListaCirmax, ListaContin, ListaDiag, ListaExpan,
ListaFladic, ListaFluxo, ListaRetira );
SetValue( Analysis:ChoiceNames, Lista_&Adic,
Lista_A&econ, Lista_&Barras, Lista_&Circex,
Lista_C&ircob, Lista_Ci&rcpl, Lista_Cir&max,
Lista_C&ontin, Lista_&Diag, Lista_E&xpan, Lista_&Fladic,
Lista_F&luxo, Lista_&Retira );
ClearList( Analysis:Commands );
SetSlotOption( Analysis:Commands, AFTER_CHANGE,
Update );
Analysis:Single = NULL;
SetSlotOption( Analysis:Single, IMAGE,
RadioButtonGroup2 );

    /*************************************
     **** INSTANCE: Default
     *************************************/
MakeInstance( Default, Root );

/************** METHOD: View  **************/
MakeMethod( Default, View, [slot View ],
  {
  If ( View #= "CASO BASE" )
   Then Display( Default:Base );
  If ( View #= ADIÇÕES )
   Then Display( Default:Adic );
  If ( View #= COORDENADAS )
   Then Display( Default:Coord );
  If ( View #= COMANDOS )
   Then Display( Default:Cmd );
  If ( View #= RESULTADOS )
   Then Display( Default:Res );
  HideImage( RadioButtonGroup5 );
  View;
  } );
MakeSlot( Default:Cmd );
SetSlotOption( Default:Cmd, INHERIT, FALSE );
Default:Cmd = comando.dat;
SetSlotOption( Default:Cmd, IF_NEEDED, NULL );
SetSlotOption( Default:Cmd, WHEN_ACCESS, NULL );
SetSlotOption( Default:Cmd, BEFORE_CHANGE, NULL );
SetSlotOption( Default:Cmd, AFTER_CHANGE, NULL );
MakeSlot( Default:Res );
SetSlotOption( Default:Res, INHERIT, FALSE );
Default:Res = result.out;
SetSlotOption( Default:Res, IF_NEEDED, NULL );
SetSlotOption( Default:Res, WHEN_ACCESS, NULL );
SetSlotOption( Default:Res, BEFORE_CHANGE, NULL );
SetSlotOption( Default:Res, AFTER_CHANGE, NULL );
MakeSlot( Default:Base );
SetSlotOption( Default:Base, INHERIT, FALSE );
Default:Base = NULL;
SetSlotOption( Default:Base, IF_NEEDED, NULL );
SetSlotOption( Default:Base, WHEN_ACCESS, NULL );
SetSlotOption( Default:Base, BEFORE_CHANGE, NULL );
SetSlotOption( Default:Base, AFTER_CHANGE, NULL );
MakeSlot( Default:Adic );
SetSlotOption( Default:Adic, INHERIT, FALSE );
Default:Adic = NULL;
SetSlotOption( Default:Adic, IF_NEEDED, NULL );
SetSlotOption( Default:Adic, WHEN_ACCESS, NULL );
SetSlotOption( Default:Adic, BEFORE_CHANGE, NULL );
SetSlotOption( Default:Adic, AFTER_CHANGE, NULL );
MakeSlot( Default:Prog );
SetSlotOption( Default:Prog, INHERIT, FALSE );
Default:Prog = sintra2.exe;
SetSlotOption( Default:Prog, IF_NEEDED, NULL );
SetSlotOption( Default:Prog, WHEN_ACCESS, NULL );
SetSlotOption( Default:Prog, BEFORE_CHANGE, NULL );
SetSlotOption( Default:Prog, AFTER_CHANGE, NULL );
MakeSlot( Default:Title );
SetSlotOption( Default:Title, INHERIT, FALSE );
Default:Title = SAPIENS;
SetSlotOption( Default:Title, IF_NEEDED, NULL );
SetSlotOption( Default:Title, WHEN_ACCESS, NULL );
SetSlotOption( Default:Title, BEFORE_CHANGE, NULL );
SetSlotOption( Default:Title, AFTER_CHANGE, NULL );
MakeSlot( Default:Start );
SetSlotOption( Default:Start, INHERIT, FALSE );
SetSlotOption( Default:Start, ALLOWABLE_VALUES, ON,
OFF );
Default:Start = OFF;
SetSlotOption( Default:Start, IF_NEEDED, NULL );
SetSlotOption( Default:Start, WHEN_ACCESS, NULL );
SetSlotOption( Default:Start, BEFORE_CHANGE, NULL );
SetSlotOption( Default:Start, AFTER_CHANGE, NULL );
MakeSlot( Default:Coord );
SetSlotOption( Default:Coord, INHERIT, FALSE );
Default:Coord = NULL;
SetSlotOption( Default:Coord, IF_NEEDED, NULL );
SetSlotOption( Default:Coord, WHEN_ACCESS, NULL );
SetSlotOption( Default:Coord, BEFORE_CHANGE, NULL );
SetSlotOption( Default:Coord, AFTER_CHANGE, NULL );
MakeSlot( Default:View );
SetSlotOption( Default:View, INHERIT, FALSE );
Default:View = CANCELA;
SetSlotOption( Default:View, IF_NEEDED, NULL );
SetSlotOption( Default:View, WHEN_ACCESS, NULL );
SetSlotOption( Default:View, BEFORE_CHANGE, View );
SetSlotOption( Default:View, AFTER_CHANGE, NULL );
SetSlotOption( Default:View, IMAGE, RadioButtonGroup5 );
```

```
    /************************************
     **** INSTANCE: Session1
     ************************************/
MakeInstance( Session1, KSession );
Session1:X = -12;
Session1:Y = 5;
Session1:Title = Session1;
Session1:SessionNumber = 1;
Session1:Width = 808;
Session1:Height = 577;
Session1:Visible = FALSE;
Session1:State = HIDDEN;
Session1:Menu = TRUE;
Session1:Titlebar = TRUE;
Session1:Sizebox = TRUE;
Session1:FileName = NULL;
SetValue( Session1:BackgroundColor, 0, 0, 255 );
Session1:RightAction = Work;
 ResetWindow ( Session1 );

    /************************************
     **** INSTANCE: View
     ************************************/
MakeInstance( View, FirstLevel );

/************ METHOD: Base  ************/
MakeMethod( View, Base, [],
  Display( Default:Base ) );

/************ METHOD: Adic  ************/
MakeMethod( View, Adic, [],
  Display( Default:Adic ) );

/************ METHOD: Command  ************/
MakeMethod( View, Command, [],
  Display( Default:Cmd ) );

/************ METHOD: Result  ************/
MakeMethod( View, Result, [],
  Display( Default:Res ) );
SetValue( View:Choices, Base, Adic, Command, Result );
SetValue( View:ChoiceNames, &Base_File, &Adic_File,
&Command_File, &Result_File );

    /************************************
     **** INSTANCE: Session2
     ************************************/
MakeInstance( Session2, KSession );
Session2:X = 93;
Session2:Y = 79;
Session2:Title = Session2;
Session2:SessionNumber = 2;
Session2:Width = 648;
Session2:Height = 332;
Session2:Visible = FALSE;
Session2:State = HIDDEN;
Session2:Menu = TRUE;
Session2:Action = Close;
Session2:Titlebar = TRUE;
Session2:Sizebox = TRUE;
Session2:Freeze = FALSE;
Session2:RightAction = Close;
SetValue( Session2:BackgroundColor, 0, 0, 255 );
 ResetWindow ( Session2 );

    /************************************
     **** INSTANCE: Study
     ************************************/
MakeInstance( Study, Root );

/************ METHOD: UpdateYearData  ************/
MakeMethod( Study, UpdateYearData, [],
  If Not( Null?( Study:Present ) )
   Then {
      HideImage( ComboBox1 );
      HideImage( RadioButtonGroup1 );
      YearData( Default:Base );
      Bus_Status( );
      General:Status = TRUE;
      Let [year Study:Present]
        If ( year > 80 )
          Then ( Text3:Title = 1900 + year )
          Else Text3:Title = 2000 + year;
      AppendToList( Commands, Commands, "PROX " #
Text3:Title );
      ShowImage( Text3 );
      SendMessage( Commands, Show );
      } );

/************ METHOD: UpdateCircuit  ************/
MakeMethod( Study, UpdateCircuit, [slot new_value ],
  {
  SetValue( new_value:Color, 2 );
  Text5:Title = "ADIC " # new_value:InitBus:Number # / #
new_value:EndBus:Number;
  ShowImage( Text5 );
  ShowImage( Button4 );
  ShowImage( Drawing1 );
  new_value;
  } );
MakeSlot( Study:Years );
SetSlotOption( Study:Years, INHERIT, FALSE );
SetSlotOption( Study:Years, MULTIPLE );
SetSlotOption( Study:Years, VALUE_TYPE, NUMBER );
ClearList( Study:Years );
```

```
SetSlotOption( Study:Years, IF_NEEDED, NULL );
SetSlotOption( Study:Years, WHEN_ACCESS, NULL );
SetSlotOption( Study:Years, BEFORE_CHANGE, NULL );
SetSlotOption( Study:Years, AFTER_CHANGE, NULL );
MakeSlot( Study:Status );
SetSlotOption( Study:Status, INHERIT, FALSE );
SetSlotOption( Study:Status, ALLOWABLE_VALUES,
NORMAL, OVERLOAD );
Study:Status = NORMAL;
SetSlotOption( Study:Status, IF_NEEDED, NULL );
SetSlotOption( Study:Status, WHEN_ACCESS, NULL );
SetSlotOption( Study:Status, BEFORE_CHANGE, NULL );
SetSlotOption( Study:Status, AFTER_CHANGE, NULL );
MakeSlot( Study:Present );
SetSlotOption( Study:Present, INHERIT, FALSE );
SetSlotOption( Study:Present, ALLOWABLE_VALUES, 93,
94, 95, 96, 97 );
Study:Present = NULL;
SetSlotOption( Study:Present, IF_NEEDED, NULL );
SetSlotOption( Study:Present, WHEN_ACCESS, NULL );
SetSlotOption( Study:Present, BEFORE_CHANGE, NULL );
SetSlotOption( Study:Present, AFTER_CHANGE,
UpdateYearData );
SetSlotOption( Study:Present, IMAGE, ComboBox1 );
MakeSlot( Study:Emergency );
SetSlotOption( Study:Emergency, INHERIT, FALSE );
SetSlotOption( Study:Emergency, ALLOWABLE_VALUES,
NORMAL, OVERLOAD );
Study:Emergency = NORMAL;
SetSlotOption( Study:Emergency, IF_NEEDED, NULL );
SetSlotOption( Study:Emergency, WHEN_ACCESS, NULL );
SetSlotOption( Study:Emergency, BEFORE_CHANGE, NULL
);
SetSlotOption( Study:Emergency, AFTER_CHANGE, NULL
);
MakeSlot( Study:Path );
SetSlotOption( Study:Path, INHERIT, FALSE );
SetSlotOption( Study:Path, VALUE_TYPE, BOOLEAN );
Study:Path = TRUE;
SetSlotOption( Study:Path, IF_NEEDED, NULL );
SetSlotOption( Study:Path, WHEN_ACCESS, NULL );
SetSlotOption( Study:Path, BEFORE_CHANGE, NULL );
SetSlotOption( Study:Path, AFTER_CHANGE, NULL );
MakeSlot( Study:Overload );
SetSlotOption( Study:Overload, INHERIT, FALSE );
SetSlotOption( Study:Overload, VALUE_TYPE, NUMBER );
Study:Overload = 20748.9962310791;
SetSlotOption( Study:Overload, IF_NEEDED, NULL );
SetSlotOption( Study:Overload, WHEN_ACCESS, NULL );
SetSlotOption( Study:Overload, BEFORE_CHANGE, NULL );
SetSlotOption( Study:Overload, AFTER_CHANGE, NULL );
MakeSlot( Study:Adic );
SetSlotOption( Study:Adic, INHERIT, FALSE );
SetSlotOption( Study:Adic, VALUE_TYPE, OBJECT );
SetSlotOption( Study:Adic, ALLOWABLE_CLASSES, LineP,
TransformerP, Line, Transformer );
Study:Adic = Line_404_412;
SetSlotOption( Study:Adic, IF_NEEDED, NULL );
SetSlotOption( Study:Adic, WHEN_ACCESS, NULL );
SetSlotOption( Study:Adic, BEFORE_CHANGE,
UpdateCircuit );
SetSlotOption( Study:Adic, AFTER_CHANGE, NULL );
SetSlotOption( Study:Adic, IMAGE, ComboBox2,
ComboBox3 );
MakeSlot( Study:Ret );
Study:Ret = Line_404_412;

    /*************************************
     **** INSTANCE: Manual
     *************************************/
MakeInstance( Manual, Synth );

/************** METHOD: Check  **************/
MakeMethod( Manual, Check, [slot Single ],
  {
  If ( Single #= "ADICIONA CIRCUITOS" )
   Then SendMessage( Self, Adic );
  If ( Single #= "RETIRA CIRCUITOS" )
   Then SendMessage( Self, Retira );
  If ( Single #= CANCELA )
   Then HideImage( RadioButtonGroup7 );
  Single;
  } );

/************** METHOD: Adic  **************/
MakeMethod( Manual, Adic, [],
  {
  ShowImage( Text4 );
  ShowImage( Text5 );
  ShowImage( ComboBox2 );
  ShowImage( Button4 );
  ShowImage( Button5 );
  } );

/************** METHOD: Retira  **************/
MakeMethod( Manual, Retira, [],
  {
  ClearList( ComboBox3:AllowableValues );
  ForAll [ x|Network ]
      If ( ( x:Ncir > 0 ) And Member?( x:Years, Text3:Title ) )
       Then AppendToList( ComboBox3:AllowableValues,
x );
  If ( LengthList( ComboBox3:AllowableValues ) #= 0 )
   Then PostMessage( "Não há circuitos a excluir neste
estágio de planejamento." )
   Else {
```

```
        ShowImage( Text7 );
        ShowImage( Text8 );
        ShowImage( ComboBox3 );
        ShowImage( Button9 );
        ShowImage( Button10 );
        };
    } );

/************ METHOD: UpdateAdic  ************/
MakeMethod( Manual, UpdateAdic, [slot new_value ],
    {
    new_value:Color = 2;
    new_value:Width = 1;
    If IsAKindOf?( new_value, Base )
     Then SetValue( new_value:Color, 3 );
    Text5:Title = "ADIC " # new_value:InitBus:Number # / #
new_value:EndBus:Number;
    ShowImage( Text5 );
    ShowImage( Drawing1 );
    new_value;
    } );

/************ METHOD: UpdateRet  ************/
MakeMethod( Manual, UpdateRet, [slot new_value ],
    {
    new_value:Color = 7;
    new_value:Width = 1;
    Text8:Title = "RETIRA " # new_value:InitBus:Number # / #
new_value:EndBus:Number;
    ShowImage( Text8 );
    ShowImage( Drawing1 );
    new_value;
    } );
MakeSlot( Manual:Adic );
SetSlotOption( Manual:Adic, INHERIT, FALSE );
SetSlotOption( Manual:Adic, VALUE_TYPE, OBJECT );
SetSlotOption( Manual:Adic, ALLOWABLE_CLASSES, Base,
Planned );
Manual:Adic = TrP_431_411;
SetSlotOption( Manual:Adic, IF_NEEDED, NULL );
SetSlotOption( Manual:Adic, WHEN_ACCESS, NULL );
SetSlotOption( Manual:Adic, BEFORE_CHANGE,
UpdateAdic );
SetSlotOption( Manual:Adic, AFTER_CHANGE, NULL );
SetSlotOption( Manual:Adic, IMAGE, ComboBox2 );
MakeSlot( Manual:Ret );
SetSlotOption( Manual:Ret, INHERIT, FALSE );
Manual:Ret = Tr_413_433;
SetSlotOption( Manual:Ret, IF_NEEDED, NULL );
SetSlotOption( Manual:Ret, WHEN_ACCESS, NULL );
SetSlotOption( Manual:Ret, BEFORE_CHANGE, UpdateRet
);
SetSlotOption( Manual:Ret, AFTER_CHANGE, NULL );
SetSlotOption( Manual:Ret, IMAGE, ComboBox3 );
Manual:Single = CANCELA;
SetSlotOption( Manual:Single, IMAGE, RadioButtonGroup7
);

    /************************************
     **** INSTANCE: Automatic
     ************************************/
MakeInstance( Automatic, Synth );

/************ METHOD: Check  ************/
MakeMethod( Automatic, Check, [slot Single ],
  {
  If ( Single #= CANCELA )
   Then HideImage( RadioButtonGroup8 )
   Else {
      If ( Single #= "ADICIONA CIRCUITOS" )
       Then {
          Self:Mode = "ADIC *";
          Text6:Title = "Adição Automática";
          };
      If ( Single #= "RETIRA CIRCUITOS" )
       Then {
          Self:Mode = "RETIRA *";
          Text6:Title = "Retirada Automática";
          };
      Show_Auto( );
      SendMessage( Self, Auto, Auto, 10 );
      };
  Single;
  } );

/************ METHOD: Auto  ************/
MakeMethod( Automatic, Auto, [slot Auto ],
  {
  Text5:Title = Self:Mode # Floor( Auto );
  ShowImage( Text5 );
  Auto;
  } );
MakeSlot( Automatic:Auto );
SetSlotOption( Automatic:Auto, INHERIT, FALSE );
Automatic:Auto = 9.99999;
SetSlotOption( Automatic:Auto, IF_NEEDED, NULL );
SetSlotOption( Automatic:Auto, WHEN_ACCESS, NULL );
SetSlotOption( Automatic:Auto, BEFORE_CHANGE, Auto );
SetSlotOption( Automatic:Auto, AFTER_CHANGE, NULL );
SetSlotOption( Automatic:Auto, IMAGE, Slider1 );
Automatic:Single = CANCELA;
SetSlotOption( Automatic:Single, IMAGE,
RadioButtonGroup8 );
MakeSlot( Automatic:Mode );
Automatic:Mode = "RETIRA *";
```

```
    /*************************************
     **** INSTANCE: ComboBox1
     *************************************/
MakeInstance( ComboBox1, ComboBox );
ComboBox1:SessionNumber = 1;
ComboBox1:Title = NULL;
SetValue( ComboBox1:ForegroundColor, 0, 0, 255 );
SetValue( ComboBox1:BackgroundColor, 255, 255, 255 );
SetValue( ComboBox1:ForegroundColor2, 0, 0, 0 );
SetValue( ComboBox1:BackgroundColor2, 255, 255, 255 );
ComboBox1:TabStop = 0;
ComboBox1:Width = 90;
ComboBox1:Height = 31;
ComboBox1:X = 624;
ComboBox1:Y = 100;
ComboBox1:Visible = FALSE;
ComboBox1:ComboEdit = FALSE;
ComboBox1:Value = NULL;
ComboBox1:Font = Arial;
ComboBox1:TextSize = 12;
ComboBox1:Bold = TRUE;
ComboBox1:Underline = FALSE;
ComboBox1:Italic = FALSE;
ComboBox1:StrikeOut = FALSE;
ComboBox1:VectorFont = RadioButtonGroup;
ComboBox1:ShowBorder = TRUE;
ComboBox1:Font2 = Arial;
ComboBox1:TextSize2 = 12;
ComboBox1:Underline2 = FALSE;
ComboBox1:Bold2 = TRUE;
ComboBox1:Italic2 = FALSE;
ComboBox1:TextVector2 = RadioButtonGroup;
ComboBox1:StrikeOut2 = FALSE;
ComboBox1:Sort = FALSE;
ComboBox1:Owner = Study;
ComboBox1:OwnerSlot = Present;
 ResetImage ( ComboBox1 );

    /*************************************
     **** INSTANCE: ComboBox2
     *************************************/
MakeInstance( ComboBox2, ComboBox );
ComboBox2:SessionNumber = 1;
ComboBox2:Title = NULL;
SetValue( ComboBox2:ForegroundColor, 0, 0, 255 );
SetValue( ComboBox2:BackgroundColor, 192, 192, 192 );
SetValue( ComboBox2:ForegroundColor2, 0, 0, 0 );
SetValue( ComboBox2:BackgroundColor2, 255, 255, 255 );
ComboBox2:TabStop = 0;
ComboBox2:Width = 133;
ComboBox2:Height = 29;
ComboBox2:X = 564;
ComboBox2:Y = 251;
ComboBox2:Visible = FALSE;
ComboBox2:Owner = Manual;
ComboBox2:OwnerSlot = Adic;
ComboBox2:Value = TrP_431_411;
ComboBox2:Font = Arial;
ComboBox2:TextSize = 12;
ComboBox2:Bold = TRUE;
ComboBox2:Underline = FALSE;
ComboBox2:Italic = FALSE;
ComboBox2:StrikeOut = FALSE;
ComboBox2:VectorFont = RadioButtonGroup;
ComboBox2:ShowBorder = FALSE;
ComboBox2:Sort = FALSE;
ComboBox2:Font2 = Arial;
ComboBox2:TextSize2 = 12;
ComboBox2:Bold2 = TRUE;
ComboBox2:Underline2 = FALSE;
ComboBox2:Italic2 = FALSE;
ComboBox2:StrikeOut2 = FALSE;
ComboBox2:TextVector2 = RadioButtonGroup;
ComboBox2:ComboEdit = FALSE;
 ResetImage ( ComboBox2 );

    /*************************************
     **** INSTANCE: ComboBox3
     *************************************/
MakeInstance( ComboBox3, ComboBox );
ComboBox3:SessionNumber = 1;
ComboBox3:Title = NULL;
SetValue( ComboBox3:ForegroundColor, 0, 0, 0 );
SetValue( ComboBox3:BackgroundColor, 192, 192, 192 );
SetValue( ComboBox3:ForegroundColor2, 0, 0, 0 );
SetValue( ComboBox3:BackgroundColor2, 255, 255, 255 );
ClearList( ComboBox3:AllowableValues );
ComboBox3:TabStop = FALSE;
ComboBox3:Width = 122;
ComboBox3:Height = 27;
ComboBox3:X = 567;
ComboBox3:Y = 266;
ComboBox3:Visible = FALSE;
ComboBox3:Font = Arial;
ComboBox3:TextSize = 12;
ComboBox3:Bold = TRUE;
ComboBox3:Underline = FALSE;
ComboBox3:Italic = TRUE;
ComboBox3:StrikeOut = FALSE;
ComboBox3:VectorFont = RadioButtonGroup;
ComboBox3:Owner = Manual;
ComboBox3:OwnerSlot = Ret;
ComboBox3:Value = Tr_413_433;
ComboBox3:ComboEdit = FALSE;
ComboBox3:Font2 = Arial;
ComboBox3:TextSize2 = 12;
```

```
ComboBox3:Bold2 = TRUE;
ComboBox3:Underline2 = FALSE;
ComboBox3:Italic2 = FALSE;
ComboBox3:StrikeOut2 = FALSE;
ComboBox3:TextVector2 = RadioButtonGroup;
ResetImage ( ComboBox3 );

    /*************************************
     **** INSTANCE: Slider1
     *************************************/
MakeInstance( Slider1, Slider );
Slider1:SessionNumber = 1;
SetValue( Slider1:ForegroundColor, 0, 0, 0 );
SetValue( Slider1:BackgroundColor, 192, 192, 192 );
SetValue( Slider1:ForegroundColor2, 0, 0, 0 );
SetValue( Slider1:BackgroundColor2, 255, 255, 255 );
Slider1:Width = 134;
Slider1:Height = 79;
Slider1:Visible = FALSE;
Slider1:X = 565;
Slider1:Y = 226;
Slider1:Owner = Automatic;
Slider1:MaxValue = 100;
Slider1:MinValue = 1;
Slider1:XFormat = FormatValue ( "\%3.0f" );
Slider1:Vertical = FALSE;
Slider1:OwnerSlot = Auto;
Slider1:Interval = 1;
Slider1:ShowBorder = FALSE;
Slider1:Font2 = Arial;
Slider1:TextSize2 = 12;
Slider1:Underline2 = FALSE;
Slider1:Bold2 = TRUE;
Slider1:Italic2 = FALSE;
Slider1:TextVector2 = RIGHT;
Slider1:StrikeOut2 = FALSE;
Slider1:Value = 9.99999;
ResetImage ( Slider1 );

    /*************************************
     **** INSTANCE: RadioButtonGroup1
     *************************************/
MakeInstance( RadioButtonGroup1, RadioButtonGroup );
RadioButtonGroup1:SessionNumber = 1;
RadioButtonGroup1:Title = "Comandos Gerais";
SetValue( RadioButtonGroup1:ForegroundColor, 128, 0, 0 );
SetValue( RadioButtonGroup1:BackgroundColor, 192, 192,
192 );
SetValue( RadioButtonGroup1:ForegroundColor2, 0, 0, 0 );
SetValue( RadioButtonGroup1:BackgroundColor2, 192, 192,
192 );
RadioButtonGroup1:X = 520;
RadioButtonGroup1:Y = 71;
SetValue( RadioButtonGroup1:AllowableValues, "NOVO
ANO", "REINICIALIZA ESTUDO", "SIMULA PLANO",
CANCELA );
RadioButtonGroup1:Width = 198;
RadioButtonGroup1:Height = 122;
RadioButtonGroup1:Visible = FALSE;
RadioButtonGroup1:Font = Arial;
RadioButtonGroup1:TextSize = 12;
RadioButtonGroup1:Bold = TRUE;
RadioButtonGroup1:Underline = FALSE;
RadioButtonGroup1:Italic = TRUE;
RadioButtonGroup1:StrikeOut = FALSE;
RadioButtonGroup1:VectorFont = CurrentTopIndex;
RadioButtonGroup1:Owner = General;
RadioButtonGroup1:OwnerSlot = Single;
RadioButtonGroup1:Value = CANCELA;
RadioButtonGroup1:TabStop = 0;
RadioButtonGroup1:Font2 = Arial;
RadioButtonGroup1:TextSize2 = 9;
RadioButtonGroup1:Bold2 = TRUE;
RadioButtonGroup1:Underline2 = FALSE;
RadioButtonGroup1:Italic2 = FALSE;
RadioButtonGroup1:StrikeOut2 = FALSE;
RadioButtonGroup1:TextVector2 = CurrentTopIndex;
RadioButtonGroup1:Transparent = FALSE;
ResetImage ( RadioButtonGroup1 );

    /*************************************
     **** INSTANCE: RadioButtonGroup2
     *************************************/
MakeInstance( RadioButtonGroup2, RadioButtonGroup );
RadioButtonGroup2:Value = CANCELA;
RadioButtonGroup2:Owner = Analy;
RadioButtonGroup2:OwnerSlot = Single;
RadioButtonGroup2:X = 519;
RadioButtonGroup2:Y = 109;
RadioButtonGroup2:Width = 144;
RadioButtonGroup2:Height = 388;
RadioButtonGroup2:SessionNumber = 1;
RadioButtonGroup2:Title = "Comandos de Análise";
RadioButtonGroup2:TabStop = 0;
SetValue( RadioButtonGroup2:ForegroundColor, 128, 0, 0 );
SetValue( RadioButtonGroup2:BackgroundColor, 192, 192,
192 );
RadioButtonGroup2:Font = Arial;
RadioButtonGroup2:TextSize = 12;
RadioButtonGroup2:VectorFont = CurrentTopIndex;
SetValue( RadioButtonGroup2:ForegroundColor2, 0, 0, 0 );
SetValue( RadioButtonGroup2:BackgroundColor2, 192, 192,
192 );
RadioButtonGroup2:Font2 = Arial;
RadioButtonGroup2:TextSize2 = 9;
RadioButtonGroup2:TextVector2 = CurrentTopIndex;
```

```
RadioButtonGroup2:Visible = FALSE;
RadioButtonGroup2:ShowBorder = TRUE;
RadioButtonGroup2:Transparent = FALSE;
SetValue( RadioButtonGroup2:AllowableValues, "LISTA
ADIC", "LISTA AECON", "LISTA BARRAS", "LISTA
CIRCEX", "LISTA CIRCOB", "LISTA CIRCPL", "LISTA
CIRCMAX", "LISTA CONTIN", "LISTA DIAG", "LISTA
EXPAN", "LISTA FLADIC", "LISTA FLUXO", "LISTA
RETIRA", CANCELA );
RadioButtonGroup2:Bold = TRUE;
RadioButtonGroup2:Underline = FALSE;
RadioButtonGroup2:Italic = TRUE;
RadioButtonGroup2:StrikeOut = FALSE;
RadioButtonGroup2:Bold2 = TRUE;
RadioButtonGroup2:Underline2 = FALSE;
RadioButtonGroup2:Italic2 = FALSE;
RadioButtonGroup2:StrikeOut2 = FALSE;
ResetImage ( RadioButtonGroup2 );

    /*************************************
     **** INSTANCE: RadioButtonGroup4
     *************************************/
MakeInstance( RadioButtonGroup4, RadioButtonGroup );
RadioButtonGroup4:SessionNumber = 1;
RadioButtonGroup4:Title = "Classes de Comandos";
SetValue( RadioButtonGroup4:ForegroundColor, 128, 0, 0 );
SetValue( RadioButtonGroup4:BackgroundColor, 192, 192,
192 );
SetValue( RadioButtonGroup4:ForegroundColor2, 0, 0, 0 );
SetValue( RadioButtonGroup4:BackgroundColor2, 192, 192,
192 );
RadioButtonGroup4:X = 504;
RadioButtonGroup4:Y = 3;
SetValue( RadioButtonGroup4:AllowableValues, SÍNTESE,
ANÁLISE, GERAL, CANCELA );
RadioButtonGroup4:Width = 150;
RadioButtonGroup4:Height = 148;
RadioButtonGroup4:Visible = FALSE;
RadioButtonGroup4:Owner = Commands;
RadioButtonGroup4:OwnerSlot = Single;
RadioButtonGroup4:TabStop = 7;
RadioButtonGroup4:Value = CANCELA;
RadioButtonGroup4:Transparent = FALSE;
RadioButtonGroup4:Font = Arial;
RadioButtonGroup4:TextSize = 12;
RadioButtonGroup4:Bold = TRUE;
RadioButtonGroup4:Underline = FALSE;
RadioButtonGroup4:Italic = TRUE;
RadioButtonGroup4:StrikeOut = FALSE;
RadioButtonGroup4:VectorFont = CurrentTopIndex;
RadioButtonGroup4:Font2 = Arial;
RadioButtonGroup4:TextSize2 = 9;
RadioButtonGroup4:Bold2 = TRUE;
RadioButtonGroup4:Underline2 = FALSE;
RadioButtonGroup4:Italic2 = FALSE;
RadioButtonGroup4:StrikeOut2 = FALSE;
RadioButtonGroup4:TextVector2 = CurrentTopIndex;
ResetImage ( RadioButtonGroup4 );

    /*************************************
     **** INSTANCE: RadioButtonGroup5
     *************************************/
MakeInstance( RadioButtonGroup5, RadioButtonGroup );
RadioButtonGroup5:SessionNumber = 1;
RadioButtonGroup5:Title = "Arquivos de Trabalho";
SetValue( RadioButtonGroup5:ForegroundColor, 128, 0, 0 );
SetValue( RadioButtonGroup5:BackgroundColor, 192, 192,
192 );
SetValue( RadioButtonGroup5:ForegroundColor2, 0, 0, 0 );
SetValue( RadioButtonGroup5:BackgroundColor2, 192, 192,
192 );
RadioButtonGroup5:X = 505;
RadioButtonGroup5:Y = 4;
SetValue( RadioButtonGroup5:AllowableValues, "CASO
BASE", ADIÇÕES, COORDENADAS, COMANDOS,
RESULTADOS, CANCELA );
RadioButtonGroup5:Width = 150;
RadioButtonGroup5:Height = 174;
RadioButtonGroup5:Visible = FALSE;
RadioButtonGroup5:Owner = Default;
RadioButtonGroup5:OwnerSlot = View;
RadioButtonGroup5:Value = CANCELA;
RadioButtonGroup5:TabStop = 0;
RadioButtonGroup5:Font = Arial;
RadioButtonGroup5:TextSize = 12;
RadioButtonGroup5:Bold = TRUE;
RadioButtonGroup5:Underline = FALSE;
RadioButtonGroup5:Italic = TRUE;
RadioButtonGroup5:StrikeOut = FALSE;
RadioButtonGroup5:VectorFont = CurrentTopIndex;
RadioButtonGroup5:Font2 = Arial;
RadioButtonGroup5:TextSize2 = 9;
RadioButtonGroup5:Bold2 = TRUE;
RadioButtonGroup5:Underline2 = FALSE;
RadioButtonGroup5:Italic2 = FALSE;
RadioButtonGroup5:StrikeOut2 = FALSE;
RadioButtonGroup5:TextVector2 = CurrentTopIndex;
ResetImage ( RadioButtonGroup5 );

    /*************************************
     **** INSTANCE: RadioButtonGroup6
     *************************************/
MakeInstance( RadioButtonGroup6, RadioButtonGroup );
RadioButtonGroup6:SessionNumber = 1;
RadioButtonGroup6:Title = "Comandos de Síntese";
SetValue( RadioButtonGroup6:ForegroundColor, 128, 0, 0 );
```

```
SetValue( RadioButtonGroup6:BackgroundColor, 192, 192,
192 );
SetValue( RadioButtonGroup6:ForegroundColor2, 0, 0, 0 );
SetValue( RadioButtonGroup6:BackgroundColor2, 192, 192,
192 );
RadioButtonGroup6:X = 520;
RadioButtonGroup6:Y = 90;
SetValue( RadioButtonGroup6:AllowableValues, MANUAL,
AUTOMÁTICO, CANCELA );
RadioButtonGroup6:Width = 150;
RadioButtonGroup6:Height = 120;
RadioButtonGroup6:Visible = FALSE;
RadioButtonGroup6:Owner = Synth;
RadioButtonGroup6:OwnerSlot = Single;
RadioButtonGroup6:Value = CANCELA;
RadioButtonGroup6:TabStop = 0;
RadioButtonGroup6:Font = Arial;
RadioButtonGroup6:TextSize = 12;
RadioButtonGroup6:Bold = TRUE;
RadioButtonGroup6:Underline = FALSE;
RadioButtonGroup6:Italic = TRUE;
RadioButtonGroup6:StrikeOut = FALSE;
RadioButtonGroup6:VectorFont = CurrentTopIndex;
RadioButtonGroup6:Font2 = Arial;
RadioButtonGroup6:TextSize2 = 9;
RadioButtonGroup6:Bold2 = TRUE;
RadioButtonGroup6:Underline2 = FALSE;
RadioButtonGroup6:Italic2 = FALSE;
RadioButtonGroup6:StrikeOut2 = FALSE;
RadioButtonGroup6:TextVector2 = CurrentTopIndex;
 ResetImage ( RadioButtonGroup6 );

    /************************************
     **** INSTANCE: RadioButtonGroup7
     ************************************/
MakeInstance( RadioButtonGroup7, RadioButtonGroup );
RadioButtonGroup7:SessionNumber = 1;
RadioButtonGroup7:Title = "Síntese Manual";
SetValue( RadioButtonGroup7:ForegroundColor, 128, 0, 0 );
SetValue( RadioButtonGroup7:BackgroundColor, 192, 192,
192 );
SetValue( RadioButtonGroup7:ForegroundColor2, 0, 0, 0 );
SetValue( RadioButtonGroup7:BackgroundColor2, 192, 192,
192 );
RadioButtonGroup7:X = 536;
RadioButtonGroup7:Y = 146;
SetValue( RadioButtonGroup7:AllowableValues, "ADICIONA
CIRCUITOS", "RETIRA CIRCUITOS", CANCELA );
RadioButtonGroup7:Width = 150;
RadioButtonGroup7:Height = 120;
RadioButtonGroup7:Visible = FALSE;
RadioButtonGroup7:Font = Arial;
RadioButtonGroup7:TextSize = 12;
RadioButtonGroup7:Bold = TRUE;
RadioButtonGroup7:Underline = FALSE;
RadioButtonGroup7:Italic = TRUE;
RadioButtonGroup7:StrikeOut = FALSE;
RadioButtonGroup7:VectorFont = CurrentTopIndex;
RadioButtonGroup7:Font2 = Arial;
RadioButtonGroup7:TextSize2 = 9;
RadioButtonGroup7:Bold2 = TRUE;
RadioButtonGroup7:Underline2 = FALSE;
RadioButtonGroup7:Italic2 = FALSE;
RadioButtonGroup7:StrikeOut2 = FALSE;
RadioButtonGroup7:TextVector2 = CurrentTopIndex;
RadioButtonGroup7:Owner = Manual;
RadioButtonGroup7:OwnerSlot = Single;
RadioButtonGroup7:Value = CANCELA;
RadioButtonGroup7:TabStop = 0;
 ResetImage ( RadioButtonGroup7 );

    /************************************
     **** INSTANCE: RadioButtonGroup8
     ************************************/
MakeInstance( RadioButtonGroup8, RadioButtonGroup );
RadioButtonGroup8:SessionNumber = 1;
RadioButtonGroup8:Title = "Síntese Automática";
SetValue( RadioButtonGroup8:ForegroundColor, 128, 0, 0 );
SetValue( RadioButtonGroup8:BackgroundColor, 192, 192,
192 );
SetValue( RadioButtonGroup8:ForegroundColor2, 0, 0, 0 );
SetValue( RadioButtonGroup8:BackgroundColor2, 192, 192,
192 );
RadioButtonGroup8:X = 536;
RadioButtonGroup8:Y = 176;
SetValue( RadioButtonGroup8:AllowableValues, "ADICIONA
CIRCUITOS", "RETIRA CIRCUITOS", CANCELA );
RadioButtonGroup8:Width = 150;
RadioButtonGroup8:Height = 120;
RadioButtonGroup8:Visible = FALSE;
RadioButtonGroup8:Font = Arial;
RadioButtonGroup8:TextSize = 12;
RadioButtonGroup8:Bold = TRUE;
RadioButtonGroup8:Underline = FALSE;
RadioButtonGroup8:Italic = TRUE;
RadioButtonGroup8:StrikeOut = FALSE;
RadioButtonGroup8:VectorFont = CurrentTopIndex;
RadioButtonGroup8:Font2 = Arial;
RadioButtonGroup8:TextSize2 = 9;
RadioButtonGroup8:Bold2 = TRUE;
RadioButtonGroup8:Underline2 = FALSE;
RadioButtonGroup8:Italic2 = FALSE;
RadioButtonGroup8:StrikeOut2 = FALSE;
RadioButtonGroup8:TextVector2 = CurrentTopIndex;
RadioButtonGroup8:Owner = Automatic;
RadioButtonGroup8:OwnerSlot = Single;
```

```
RadioButtonGroup8:Value = CANCELA;
RadioButtonGroup8:TabStop = 0;
 ResetImage ( RadioButtonGroup8 );

    /*************************************
     **** INSTANCE: Button7
     *************************************/
MakeInstance( Button7, Button );
Button7:SessionNumber = 1;
Button7:Title = Sair;
SetValue( Button7:ForegroundColor, 0, 0, 0 );
SetValue( Button7:BackgroundColor, 192, 192, 192 );
SetValue( Button7:ForegroundColor2, 0, 0, 0 );
SetValue( Button7:BackgroundColor2, 255, 255, 255 );
Button7:Visible = TRUE;
Button7:X = 680;
Button7:Y = 300;
Button7:Width = 100;
Button7:Height = 25;
Button7:Action = End_Work;
Button7:Font = "MS Sans Serif";
Button7:TextSize = 8;
Button7:Bold = TRUE;
Button7:Underline = FALSE;
Button7:Italic = FALSE;
Button7:StrikeOut = FALSE;
Button7:VectorFont = DBCloseIndexFile;
Button7:TabStop = 3;
 ResetImage ( Button7 );

    /*************************************
     **** INSTANCE: Button4
     *************************************/
MakeInstance( Button4, Button );
Button4:SessionNumber = 1;
Button4:Title = OK;
SetValue( Button4:ForegroundColor, 0, 0, 0 );
SetValue( Button4:BackgroundColor, 192, 192, 192 );
SetValue( Button4:ForegroundColor2, 0, 0, 0 );
SetValue( Button4:BackgroundColor2, 255, 255, 255 );
Button4:Visible = FALSE;
Button4:X = 552;
Button4:Y = 358;
Button4:Width = 75;
Button4:Height = 25;
Button4:Action = Adic;
 ResetImage ( Button4 );

    /*************************************
     **** INSTANCE: Button5
     *************************************/
MakeInstance( Button5, Button );
Button5:SessionNumber = 1;
Button5:Title = CANCEL;
SetValue( Button5:ForegroundColor, 0, 0, 0 );
SetValue( Button5:BackgroundColor, 192, 192, 192 );
SetValue( Button5:ForegroundColor2, 0, 0, 0 );
SetValue( Button5:BackgroundColor2, 255, 255, 255 );
Button5:Visible = FALSE;
Button5:X = 627;
Button5:Y = 358;
Button5:Width = 75;
Button5:Height = 25;
Button5:Action = Cancel_Adic;
 ResetImage ( Button5 );

    /*************************************
     **** INSTANCE: Button6
     *************************************/
MakeInstance( Button6, Button );
Button6:SessionNumber = 1;
Button6:Title = "Novo Estudo";
SetValue( Button6:ForegroundColor, 0, 0, 0 );
SetValue( Button6:BackgroundColor, 192, 192, 192 );
SetValue( Button6:ForegroundColor2, 0, 0, 0 );
SetValue( Button6:BackgroundColor2, 255, 255, 255 );
Button6:Visible = TRUE;
Button6:X = 680;
Button6:Y = 156;
Button6:Action = Start;
Button6:Width = 100;
Button6:Height = 25;
 ResetImage ( Button6 );

    /*************************************
     **** INSTANCE: Button1
     *************************************/
MakeInstance( Button1, Button );
Button1:SessionNumber = 1;
Button1:Title = "Fluxo Pot.";
SetValue( Button1:ForegroundColor, 0, 0, 0 );
SetValue( Button1:BackgroundColor, 192, 192, 192 );
SetValue( Button1:ForegroundColor2, 0, 0, 0 );
SetValue( Button1:BackgroundColor2, 255, 255, 255 );
Button1:Visible = FALSE;
Button1:X = 680;
Button1:Y = 260;
Button1:Width = 100;
Button1:Height = 25;
Button1:Action = Overload;
 ResetImage ( Button1 );

    /*************************************
     **** INSTANCE: Button2
     *************************************/
MakeInstance( Button2, Button );
```

```
Button2:SessionNumber = 1;
Button2:Title = Consulta;
SetValue( Button2:ForegroundColor, 0, 0, 0 );
SetValue( Button2:BackgroundColor, 192, 192, 192 );
SetValue( Button2:ForegroundColor2, 0, 0, 0 );
SetValue( Button2:BackgroundColor2, 255, 255, 255 );
Button2:Visible = FALSE;
Button2:X = 680;
Button2:Y = 225;
Button2:Action = View;
Button2:Width = 100;
Button2:Height = 25;
 ResetImage ( Button2 );

    /*************************************
     **** INSTANCE: Button8
     *************************************/
MakeInstance( Button8, Button );
Button8:SessionNumber = 1;
Button8:Title = Comandos;
SetValue( Button8:ForegroundColor, 0, 0, 0 );
SetValue( Button8:BackgroundColor, 192, 192, 192 );
SetValue( Button8:ForegroundColor2, 0, 0, 0 );
SetValue( Button8:BackgroundColor2, 255, 255, 255 );
Button8:Visible = FALSE;
Button8:X = 680;
Button8:Y = 190;
Button8:Width = 100;
Button8:Height = 25;
Button8:Action = Comandos;
 ResetImage ( Button8 );

    /*************************************
     **** INSTANCE: Button9
     *************************************/
MakeInstance( Button9, Button );
Button9:SessionNumber = 1;
Button9:Title = OK;
SetValue( Button9:ForegroundColor, 0, 0, 0 );
SetValue( Button9:BackgroundColor, 192, 192, 192 );
SetValue( Button9:ForegroundColor2, 0, 0, 0 );
SetValue( Button9:BackgroundColor2, 255, 255, 255 );
Button9:Visible = FALSE;
Button9:X = 552;
Button9:Y = 366;
Button9:Action = Ret;
Button9:Width = 75;
Button9:Height = 25;
 ResetImage ( Button9 );

    /*************************************
     **** INSTANCE: Button10
     *************************************/
MakeInstance( Button10, Button );
Button10:SessionNumber = 1;
Button10:Title = CANCEL;
SetValue( Button10:ForegroundColor, 0, 0, 0 );
SetValue( Button10:BackgroundColor, 192, 192, 192 );
SetValue( Button10:ForegroundColor2, 0, 0, 0 );
SetValue( Button10:BackgroundColor2, 255, 255, 255 );
Button10:Visible = FALSE;
Button10:X = 625;
Button10:Y = 366;
Button10:Action = Cancel_Ret;
Button10:Height = 25;
Button10:Width = 75;
 ResetImage ( Button10 );

    /*************************************
     **** INSTANCE: Button3
     *************************************/
MakeInstance( Button3, Button );
Button3:SessionNumber = 1;
Button3:Title = OK;
SetValue( Button3:ForegroundColor, 0, 0, 0 );
SetValue( Button3:BackgroundColor, 192, 192, 192 );
SetValue( Button3:ForegroundColor2, 0, 0, 0 );
SetValue( Button3:BackgroundColor2, 255, 255, 255 );
Button3:Visible = FALSE;
Button3:X = 550;
Button3:Y = 359;
Button3:Width = 79;
Button3:Height = 25;
Button3:Action = Auto;
 ResetImage ( Button3 );

    /*************************************
     **** INSTANCE: Button11
     *************************************/
MakeInstance( Button11, Button );
Button11:SessionNumber = 1;
Button11:Title = CANCEL;
SetValue( Button11:ForegroundColor, 0, 0, 0 );
SetValue( Button11:BackgroundColor, 192, 192, 192 );
SetValue( Button11:ForegroundColor2, 0, 0, 0 );
SetValue( Button11:BackgroundColor2, 255, 255, 255 );
Button11:Visible = FALSE;
Button11:X = 627;
Button11:Y = 359;
Button11:Width = 79;
Button11:Height = 25;
Button11:Action = Cancel_Auto;
 ResetImage ( Button11 );

    /*************************************
     **** INSTANCE: Text1
```

```
    ************************************/
MakeInstance( Text1, Text );
Text1:SessionNumber = 1;
Text1:Title = "DIAGRAMA UNIFILAR";
SetValue( Text1:ForegroundColor, 255, 255, 255 );
SetValue( Text1:BackgroundColor, 0, 0, 255 );
SetValue( Text1:ForegroundColor2, 0, 0, 0 );
SetValue( Text1:BackgroundColor2, 255, 255, 255 );
Text1:Width = 137;
Text1:Height = 16;
Text1:X = 8;
Text1:Y = 471;
Text1:Visible = FALSE;
Text1:Font = Arial;
Text1:TextSize = 12;
Text1:Bold = TRUE;
Text1:Underline = FALSE;
Text1:Italic = FALSE;
Text1:StrikeOut = FALSE;
Text1:VectorFont = PrintImage;
Text1:ShowBorder = FALSE;
Text1:Justification = LEFT;
 ResetImage ( Text1 );

    /************************************
    **** INSTANCE: Text2
    ************************************/
MakeInstance( Text2, Text );
Text2:SessionNumber = 1;
Text2:Title = "ANO EM ESTUDO";
SetValue( Text2:ForegroundColor, 255, 255, 255 );
SetValue( Text2:BackgroundColor, 0, 0, 255 );
SetValue( Text2:ForegroundColor2, 0, 0, 0 );
SetValue( Text2:BackgroundColor2, 255, 255, 255 );
Text2:Width = 114;
Text2:Height = 16;
Text2:X = 8;
Text2:Y = 494;
Text2:Visible = FALSE;
Text2:Font = Arial;
Text2:TextSize = 12;
Text2:Bold = TRUE;
Text2:Underline = FALSE;
Text2:Italic = FALSE;
Text2:StrikeOut = FALSE;
Text2:VectorFont = PrintImage;
Text2:ShowBorder = FALSE;
Text2:Justification = LEFT;
 ResetImage ( Text2 );

    /************************************
    **** INSTANCE: Text3
    ************************************/
MakeInstance( Text3, Text );
Text3:SessionNumber = 1;
Text3:Title = 1993;
SetValue( Text3:ForegroundColor, 128, 0, 0 );
SetValue( Text3:BackgroundColor, 192, 192, 192 );
SetValue( Text3:ForegroundColor2, 0, 0, 0 );
SetValue( Text3:BackgroundColor2, 255, 255, 255 );
Text3:Width = 50;
Text3:Height = 20;
Text3:X = 127;
Text3:Y = 493;
Text3:Visible = FALSE;
Text3:Font = Arial;
Text3:TextSize = 12;
Text3:Bold = TRUE;
Text3:Underline = FALSE;
Text3:Italic = TRUE;
Text3:StrikeOut = FALSE;
Text3:VectorFont = PrintImage;
Text3:Justification = CENTER;
Text3:ShowBorder = TRUE;
 ResetImage ( Text3 );

    /************************************
    **** INSTANCE: Text5
    ************************************/
MakeInstance( Text5, Text );
Text5:SessionNumber = 1;
Text5:Title = "ADIC 431/411";
SetValue( Text5:ForegroundColor, 0, 0, 128 );
SetValue( Text5:BackgroundColor, 192, 192, 192 );
SetValue( Text5:ForegroundColor2, 0, 0, 0 );
SetValue( Text5:BackgroundColor2, 255, 255, 255 );
Text5:Width = 133;
Text5:Height = 22;
Text5:X = 565;
Text5:Y = 314;
Text5:Visible = FALSE;
Text5:ShowBorder = FALSE;
Text5:Justification = CENTER;
Text5:Font = "MS Sans Serif";
Text5:TextSize = 12;
Text5:Bold = TRUE;
Text5:Underline = FALSE;
Text5:Italic = TRUE;
Text5:StrikeOut = FALSE;
Text5:VectorFont = PrintImage;
 ResetImage ( Text5 );

    /************************************
    **** INSTANCE: Text4
    ************************************/
MakeInstance( Text4, Text );
```

```
Text4:SessionNumber = 1;
Text4:Title = "Escolha Circuito a Adicionar";
SetValue( Text4:ForegroundColor, 128, 0, 0 );
SetValue( Text4:BackgroundColor, 192, 192, 192 );
SetValue( Text4:ForegroundColor2, 0, 0, 0 );
SetValue( Text4:BackgroundColor2, 255, 255, 255 );
Text4:Width = 150;
Text4:Height = 182;
Text4:X = 552;
Text4:Y = 201;
Text4:Visible = FALSE;
Text4:Font = Arial;
Text4:TextSize = 12;
Text4:Bold = TRUE;
Text4:Underline = FALSE;
Text4:Italic = TRUE;
Text4:StrikeOut = FALSE;
Text4:VectorFont = PrintImage;
Text4:ShowBorder = TRUE;
Text4:Justification = CENTER;
ResetImage ( Text4 );

    /*************************************
     **** INSTANCE: Text6
     *************************************/
MakeInstance( Text6, Text );
Text6:SessionNumber = 1;
Text6:Title = "Retirada Automática";
SetValue( Text6:ForegroundColor, 128, 0, 0 );
SetValue( Text6:BackgroundColor, 192, 192, 192 );
SetValue( Text6:ForegroundColor2, 0, 0, 0 );
SetValue( Text6:BackgroundColor2, 255, 255, 255 );
Text6:Width = 158;
Text6:Height = 182;
Text6:X = 549;
Text6:Y = 203;
Text6:Visible = FALSE;
Text6:Font = Arial;
Text6:TextSize = 12;
Text6:Bold = TRUE;
Text6:Underline = FALSE;
Text6:Italic = TRUE;
Text6:StrikeOut = FALSE;
Text6:VectorFont = PrintImage;
Text6:ShowBorder = TRUE;
Text6:Justification = CENTER;
ResetImage ( Text6 );

    /*************************************
     **** INSTANCE: Text8
     *************************************/
MakeInstance( Text8, Text );
Text8:SessionNumber = 1;
Text8:Title = "RETIRA 413/433";
SetValue( Text8:ForegroundColor, 0, 0, 128 );
SetValue( Text8:BackgroundColor, 192, 192, 192 );
SetValue( Text8:ForegroundColor2, 0, 0, 0 );
SetValue( Text8:BackgroundColor2, 255, 255, 255 );
Text8:Width = 130;
Text8:Height = 25;
Text8:X = 563;
Text8:Y = 312;
Text8:Visible = FALSE;
Text8:Font = Arial;
Text8:TextSize = 14;
Text8:Bold = TRUE;
Text8:Underline = FALSE;
Text8:Italic = TRUE;
Text8:StrikeOut = FALSE;
Text8:VectorFont = PrintImage;
Text8:Justification = CENTER;
ResetImage ( Text8 );

    /*************************************
     **** INSTANCE: Text9
     *************************************/
MakeInstance( Text9, Text );
Text9:SessionNumber = 1;
Text9:Title = "Rede com Barras Desconectadas";
SetValue( Text9:ForegroundColor, 128, 0, 0 );
SetValue( Text9:BackgroundColor, 192, 192, 192 );
SetValue( Text9:ForegroundColor2, 0, 0, 0 );
SetValue( Text9:BackgroundColor2, 255, 255, 255 );
Text9:Width = 268;
Text9:Height = 22;
Text9:X = 234;
Text9:Y = 467;
Text9:Visible = FALSE;
Text9:Font = Arial;
Text9:TextSize = 12;
Text9:Bold = TRUE;
Text9:Underline = FALSE;
Text9:Italic = FALSE;
Text9:StrikeOut = FALSE;
Text9:VectorFont = PrintImage;
Text9:Justification = CENTER;
Text9:ShowBorder = TRUE;
ResetImage ( Text9 );

    /*************************************
     **** INSTANCE: Text10
     *************************************/
MakeInstance( Text10, Text );
Text10:SessionNumber = 1;
Text10:Title = "Rede com Sobrecarga";
SetValue( Text10:ForegroundColor, 128, 0, 0 );
```

```
SetValue( Text10:BackgroundColor, 192, 192, 192 );
SetValue( Text10:ForegroundColor2, 0, 0, 0 );
SetValue( Text10:BackgroundColor2, 255, 255, 255 );
Text10:Width = 267;
Text10:Height = 22;
Text10:X = 234;
Text10:Y = 490;
Text10:Visible = FALSE;
Text10:Font = Arial;
Text10:TextSize = 12;
Text10:Bold = TRUE;
Text10:Underline = FALSE;
Text10:Italic = FALSE;
Text10:StrikeOut = FALSE;
Text10:VectorFont = PrintImage;
Text10:Justification = CENTER;
Text10:ShowBorder = TRUE;
ResetImage ( Text10 );

    /************************************
     **** INSTANCE: Transcript1
     ************************************/
MakeInstance( Transcript1, Transcript );
Transcript1:SessionNumber = 2;
Transcript1:Title = NULL;
SetValue( Transcript1:ForegroundColor, 0, 0, 0 );
SetValue( Transcript1:BackgroundColor, 255, 255, 255 );
SetValue( Transcript1:ForegroundColor2, 0, 0, 0 );
SetValue( Transcript1:BackgroundColor2, 255, 255, 255 );
Transcript1:Width = 611;
Transcript1:Height = 233;
Transcript1:Visible = TRUE;
Transcript1:X = 13;
Transcript1:Y = 3;
Transcript1:VertScroll = AUTO;
Transcript1:HorzScroll = AUTO;
ResetImage ( Transcript1 );

    /************************************
     **** INSTANCE: Bitmap1
     ************************************/
MakeInstance( Bitmap1, Bitmap );
Bitmap1:SessionNumber = 1;
Bitmap1:Title = Bitmap1;
SetValue( Bitmap1:ForegroundColor, 0, 0, 0 );
SetValue( Bitmap1:BackgroundColor, 0, 0, 255 );
SetValue( Bitmap1:ForegroundColor2, 0, 0, 0 );
SetValue( Bitmap1:BackgroundColor2, 255, 255, 255 );
Bitmap1:Width = 142;
Bitmap1:Height = 127;
Bitmap1:Visible = FALSE;
Bitmap1:X = 683;
Bitmap1:Y = 3;
Bitmap1:FileName = homo.bmp;
Bitmap1:FitToScreen = TRUE;
Bitmap1:ShowBorder = FALSE;
Bitmap1:Transparent = TRUE;
ResetImage ( Bitmap1 );

    /************************************
     **** INSTANCE: Bitmap2
     ************************************/
MakeInstance( Bitmap2, Bitmap );
Bitmap2:SessionNumber = 1;
Bitmap2:Title = Bitmap2;
SetValue( Bitmap2:ForegroundColor, 0, 0, 0 );
SetValue( Bitmap2:BackgroundColor, 255, 255, 255 );
SetValue( Bitmap2:ForegroundColor2, 0, 0, 0 );
SetValue( Bitmap2:BackgroundColor2, 255, 255, 255 );
Bitmap2:Width = 541;
Bitmap2:Height = 442;
Bitmap2:Visible = TRUE;
Bitmap2:X = 110;
Bitmap2:Y = 31;
Bitmap2:FileName = homo.bmp;
Bitmap2:ShowBorder = FALSE;
ResetImage ( Bitmap2 );

    /************************************
     **** INSTANCE: Drawing1
     ************************************/
MakeInstance( Drawing1, Drawing );
Drawing1:SessionNumber = 1;
Drawing1:Title = Drawing1;
SetValue( Drawing1:ForegroundColor, 0, 0, 0 );
SetValue( Drawing1:BackgroundColor, 255, 255, 255 );
SetValue( Drawing1:ForegroundColor2, 0, 0, 0 );
SetValue( Drawing1:BackgroundColor2, 255, 255, 255 );
Drawing1:Width = 497;
Drawing1:Height = 463;
Drawing1:Visible = FALSE;
Drawing1:X = 6;
Drawing1:Y = 3;
Drawing1:Transparent = FALSE;
Drawing1:FileName = NULL;
Drawing1:XLeft = -10;
Drawing1:XRight = 150;
Drawing1:YBottom = 150;
Drawing1:YTop = -10;
Drawing1:TextSize = 12;
Drawing1:Draw = Show_System;
Drawing1:DrawFunction = Show_System;
Drawing1:Font = "Times New Roman";
Drawing1:Bold = TRUE;
Drawing1:Underline = FALSE;
Drawing1:Italic = FALSE;
```

```
Drawing1:StrikeOut = FALSE;
Drawing1:VectorFont = " ResetImage ( ";
Drawing1:MetaDraw = FALSE;
 ResetImage ( Drawing1 );



/*****************************************************/
/**       ALL RULES ARE SAVED BELOW          **/
/*****************************************************/

    /************************************
     **** RULE: Overload_Flow
     ************************************/
MakeRule( Overload_Flow, [x|Branch],
   x:Flow > x:NormalLimit,
   x:Status = OVERLOAD );
SetRulePriority( Overload_Flow, 10 );

    /************************************
     **** RULE: Normal_Flow
     ************************************/
MakeRule( Normal_Flow, [x|Branch],
   x:Flow <= x:NormalLimit,
   x:Status = NORMAL );
SetRulePriority( Normal_Flow, 10 );

    /************************************
     **** RULE: Emergency_Normal
     ************************************/
MakeRule( Emergency_Normal, [x|Branch],
   x:Flow <= x:EmergencyLimit,
   x:Emergency = NORMAL );
SetRulePriority( Emergency_Normal, 2 );

    /************************************
     **** RULE: Emergency_Over
     ************************************/
MakeRule( Emergency_Over, [x|Branch],
   x:Flow > x:EmergencyLimit,
   x:Emergency = OVERLOAD );
SetRulePriority( Emergency_Over, 2 );

    /************************************
     **** RULE: Rule1
     ************************************/
MakeRule( Rule1, [plan|Planned],
   plan:InitBus:Link #= CONNECTED And plan:EndBus:Link
#= CONNECTED,
   {
   Study:Path = TRUE;
   plan:Expansion = INSERT;
   } );



/*****************************************************/
/**       ALL GOALS ARE SAVED BELOW          **/
/*****************************************************/

    /************************************
     **** GOAL: Complete_Way
     ************************************/
MakeGoal( Complete_Way,
   Study:Way #= TRUE );
```

APÊNDICE D - FUNÇÕES DE BIBLIOTECA DEFINIDAS PELO USUÁRIO

D.1. INTRODUÇÃO

Este apêndice contém a listagem do código fonte das três rotinas pré-compiladas desenvolvidas para o protótipo SAPIENS, conhecidas por DLL - *Dynamic Link Library*. Tais rotinas ligam-se ao código executável em tempo de execução, o que otimiza o uso da memória RAM. Além disso, podem se ligar a qualquer código executável independente do compilador base adotado, facilitando a interface entre programas de diferentes linguagens como foi o caso do protótipo SAPIENS. Esta característica é muito útil em ambiente Windows.

Os arquivos necessários para construir uma DLL são:

a) udllinit.c

Um arquivo escrito em C, que acompanha o pacote KAPPA-PC. É feita uma cópia do mesmo para registrar e inicializar internamente todas as interfaces de chamadas, para indicar o mecanismo de chamada de uma função DLL definida.

b) template.def

Um arquivo com especificações de definições para o Microsoft Windows. Todas as DLL exportáveis são declaradas aqui.

c) template.mak

Um arquivo *makefile* contendo as diretivas de compilação e link-edição. Este arquivo deve ser editado para incluir os nomes apropriados das DLL criadas.

d) template.lnk

Contém os parâmetros de link-edição para o compilador C adotado.

e) user.c

Código fonte do arquivo contendo as rotinas a serem transformadas em DLL.

A Figura D.1exemplifica este processo.

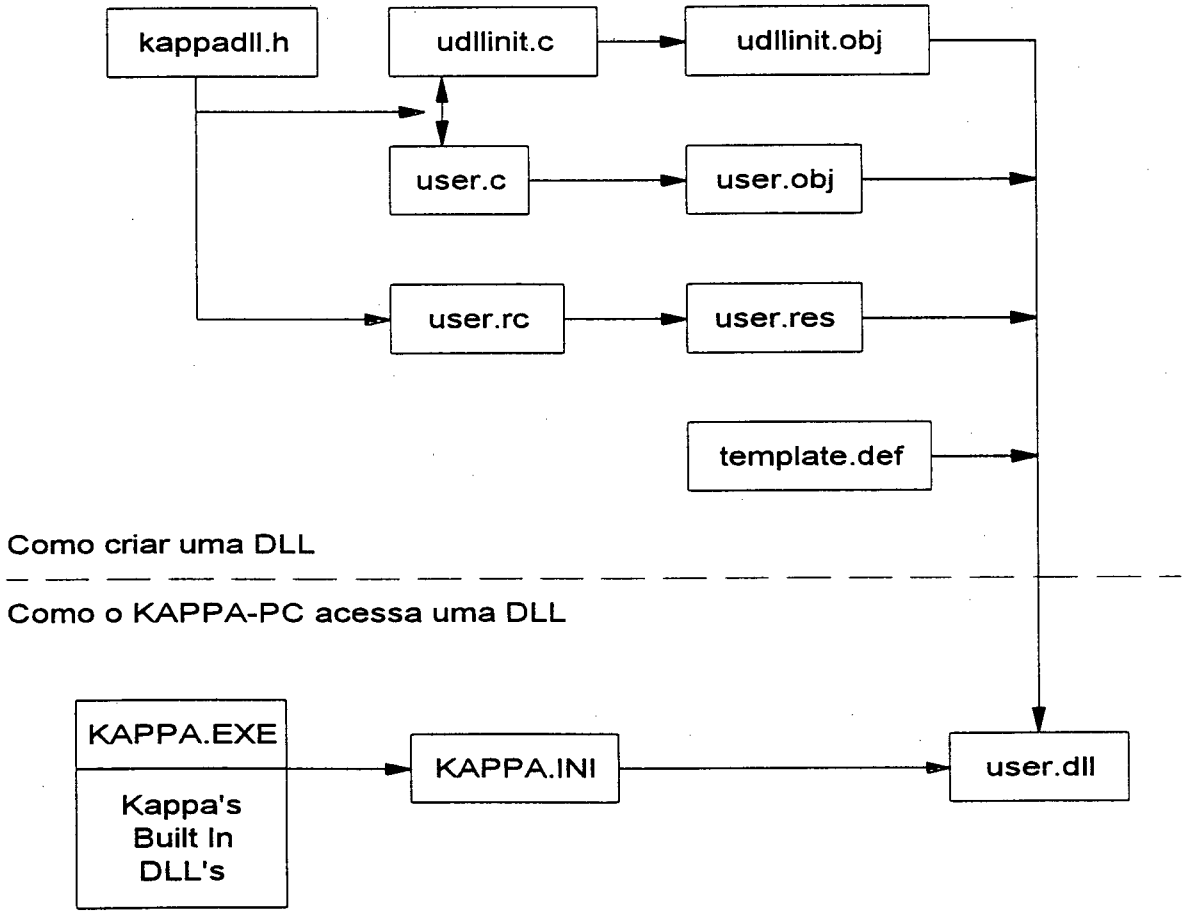


Figura D.1 - Como construir uma DLL

A seguir, a listagem do código fonte das três DLL implementadas na última versão do protótipo SAPIENS.

a) Primeira DLL - leitura dos dados dos arquivos de entrada.

A primeira DLL, efetua leitura formatada dos arquivos de dados. Assim, não foi preciso redefinir os arquivos de dados, adotando-se a mesma formatação sugerida pelo modelo SINTRA. Foi desenvolvida em linguagem C, Borland Inc., e compilada com os recursos fornecidos pelo *shell* KAPPA-PC (ver [30]).

```
/*
    DLL SapOpenFiles
    Autor: Hercídio de Paula Silveira Neto (Eng.
Eletricista)

    Parâmetros de Chamada:
    1) Nome do arquivo de dados do caso base
    2) Nome do arquivo de dados de possíveis adições

    Monta Base de Conhecimentos no KAPPA-PC
        - Bus > dados de barras;
        - Branch > dados de Trafos (1) e LT's (2)
existentes;
        - Planned > dados de circuitos a adicionar;
        - Study > lista com anos de estudo.

    Lista com rotinas Kpc e Kpa utilizadas:
    . KpaArgCount      / KpaReturnError /
KpaGetArgAtom
    . KpaReturnTrue / KpcMakeInstance / KpcSetValueF
    . _a2o, _i2a, f2a, _s2a
*/

# include "kappadll.h"
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# include <ctype.h>
# include <conio.h>

void readata ( FILE *fp );

void bus_slots ( FILE *fp );

void base_slots ( FILE *fp, int type );

void study_slots ( FILE *fp, int type );

void planned_slots ( FILE *fp, int type );

float extract ( char source[72], int begin, int end );

void bus_number ( int bus, char dest[11] );

short PEXPORT SapOpenFiles ( ARGLIST lpArgList )
{
    FILE *fp1, *fp2;
    ATOMID idAtom;
    char file_base[40], file_adic[40];

    if ( KpaArgCount ( lpArgList ) != 2 )
        KpaReturnError;
    if ( !KpaGetArgAtom ( lpArgList, 1, idAtom ) )
        KpaReturnError;
    _a2s ( file_base, idAtom );
    if ( !KpaGetArgAtom ( lpArgList, 2, idAtom ) )
        KpaReturnError;
    _a2s ( file_adic, idAtom );
    if (( fp1 = fopen ( file_base, "rt" )) == NULL)
        KpaReturnError;
    if (( fp2 = fopen ( file_adic, "rt" )) == NULL)
        KpaReturnError;
    readata ( fp1 );
    fclose ( fp1 );
    readata ( fp2 );
    fclose ( fp2 );
    KpaReturnTrue;
}

void readata ( FILE *fp )
{
    char line[72];
    char ch;

    while ( !feof ( fp ))
    {
        ch = fgetc ( fp );
        if ( ch == '*' )
        {
            fgets ( line, 10, fp );
            if ( !strchr ( line, '*' ) )
            {
                strlwr ( line );
                line[0] = _toupper ( line[0] );
                if ( strstr ( line, "Bus" ))
                    bus_slots ( fp );
                if ( strstr ( line, "Trex" ))
                    base_slots ( fp, 1 );
                if ( strstr ( line, "Ltex" ))
                    base_slots ( fp, 2 );
                if ( strstr ( line, "Gat" ))
                    study_slots ( fp, 1 );
                if ( strstr ( line, "Cat" ))
                study_slots ( fp, 2 );
                if ( strstr ( line, "Adlt" ))
                    planned_slots ( fp, 2 );
                if ( strstr ( line, "Adtr" ))
                    planned_slots ( fp, 1 );
            };
        };
    };
}

void bus_slots ( FILE *fp )
{
    ATOMID idInst, idObj;
    fpos_t filepos;
    char aux[11];
    char line[72], bname[12];
    int nbus, area, i;
    float volt;

    idObj = _s2a ( "Bus" );
    do
    {
        fgetpos ( fp, &filepos );
        fgets ( line, 72, fp );
        if ( !strchr ( line, ';' ) && !strchr ( line, '*' ) )
        {
            nbus = extract ( line, 0, 4 );
            area = extract ( line, 18, 20 );
            volt = extract ( line, 28, 33 );
```

```
            for ( i=5; i<17; i++) bname[i-5] = line[i];
            bname[12] = '\0';
            bus_number ( nbus, aux );
            idInst = _s2a ( aux );
            KpcMakeInstance ( idInst, _a2o ( idObj ) );
            KpcSetValueF ( _a2o ( idInst ), _s2a (
"Number" ), _i2a ( nbus ) );
            KpcSetValueF ( _a2o ( idInst ), _s2a ( "Name"
), _s2a ( bname ) );
            KpcSetValueF ( _a2o ( idInst ), _s2a ( "Area" ),
_i2a ( area ) );
            KpcSetValueF ( _a2o ( idInst ), _s2a (
"VoltClass" ), _f2a ( volt ) );
        };
    } while ( !strchr ( line, '*' ) );
    fsetpos ( fp, &filepos );
}

void base_slots ( FILE *fp, int type )
/*
    Type = 1 => Transformador Existente
    Type = 2 => Linha de Transmissão Existente
*/
{
    ATOMID idInst, idObj;
    fpos_t filepos;
    char aux[11], str[4];
    char line[72];
    int ibus, ebus, ncir, i;
    static int j = 1;
    float flow_lim, length;

    idObj = _s2a ( "Transformer" );
    if ( type == 2 ) idObj = _s2a ( "Line" );
    do
    {
        fgetpos ( fp, &filepos );
        fgets ( line, 72, fp );
        if ( !strchr ( line, ';' ) && !strchr ( line, '*' ) )
        {
            ibus = extract ( line, 0, 4 );
            ebus = extract ( line, 5, 9 );
            ncir = extract ( line, 10, 12 );
            flow_lim = extract ( line, 27+(type-1)*14,
32+(type-1)*14 );
            strcpy ( aux, "Tr_" );
            if ( type == 2 ) strcpy ( aux, "Line_" );
            itoa ( ibus, str, 10 );
            strcat ( aux, str );
            strcat ( aux, "_" );
            itoa ( ebus, str, 10 );
            strcat ( aux, str );
            idInst = _s2a ( aux );
            KpcMakeInstance ( idInst, _a2o ( idObj ) );
            bus_number ( ibus, aux );
            KpcSetValueF ( _a2o ( idInst ), _s2a ( "InitBus"
), _s2a (aux) );
            bus_number ( ebus, aux );
            KpcSetValueF ( _a2o ( idInst ), _s2a (
"EndBus" ), _s2a (aux) );
            KpcSetValueF ( _a2o ( idInst ), _s2a (
"Circuits" ), _i2a ( ncir ) );
            KpcSetValueF ( _a2o ( idInst ), _s2a (
"FlowLimit" ), _f2a ( flow_lim ) );
            j++;
        };
    } while ( !strchr ( line, '*' ) );
    if ( type == 2 ) j = 1;
    fsetpos ( fp, &filepos );
}

void study_slots ( FILE *fp, int type )
{
/*  Type 1 => Dados de Geracao, Anos de Estudo
    Type 2 => Dados de Demanda*/

    ATOMID idInst, idObj;
    fpos_t filepos;
    char line[80], aux[11], str[4];
    int ny, num, i, k = 1;
    static int years[20];
    float value;

    do
    {
        fgets ( line, 80, fp );
    } while ( strchr ( line, ';' ) );
    ny = extract ( line, 0, 2 );
    if ( type == 1 ) {
        idInst = _s2a ( "Study" );
        for ( i= 0; i<ny; i++)
        {
            num = extract ( line, k+2, k+4 );
            KpcInsertNthElem ( _a2o ( idInst ), _s2a (
"Years" ), i+1, _i2a ( num ));
            years[i] = num;
            k+=3;
        };
    };
    idObj = _s2a ( "Generator" );
    if ( type == 2 ) idObj = _s2a ( "Load" );
    do {
        i = 0;
        value = 0;
        fgetpos ( fp, &filepos );
        fgets ( line, 80, fp );
        if ( !strchr ( line, ';' ) && !strchr ( line, '*' )
            && !strchr ( line, '#' )) {
            num = extract ( line, 0, 4 );
            do {
                value = extract ( line, 14+8*i, 21+8*i );
                i++;
            } while (( value == 0) && ( i < ny ));
            if (( i <= ny ) && ( value != 0 )) {
                strcpy ( aux, "Gen_");
                if ( type == 2 ) strcpy ( aux, "Load_" );
                itoa ( num, str, 10 );
                strcat ( aux, str );
                idInst = _s2a ( aux );
                bus_number(num, aux);
                KpcMakeInstance ( idInst, _a2o ( idObj ));
                KpcSetValueF ( _a2o ( idInst ), _s2a (
"Year" ), _i2a (years[i-1]) );
                KpcSetValueF ( _a2o ( idInst ), _s2a (
"Number" ), _s2a (aux) );
            };
        };
    } while ( !strchr ( line, '*' ) && !strchr ( line, '#' ));
    fsetpos ( fp, &filepos );
}

void planned_slots ( FILE *fp, int type )
/*
    Type = 1 => Transformador Planejado
    Type = 2 => Linha de Transmissão Planejada
*/
{
    ATOMID idInst, idObj;
    fpos_t filepos;
    char aux[11], str[4];
    char line[72];
    int ibus, ebus, ncir, i;
    static int j=1;
    float flow_lim, length;

    idObj = _s2a ( "TransformerP" );
```

```
    if ( type == 2 ) idObj = _s2a ( "LineP" );
    do
    {
        fgetpos ( fp, &filepos );
        fgets ( line, 72, fp );
        if ( !strchr ( line, ';' ) && !strchr ( line, '*' ) )
        {
            ibus = extract ( line, 0, 4 );
            ebus = extract ( line, 5, 9 );
            ncir = extract ( line, 13+(2-type), 15+(2-type) );
            flow_lim = extract ( line, 16, 22 );
            strcpy ( aux, "TrP_" );
            if ( type == 2 ) strcpy ( aux, "LineP_" );
            itoa ( ibus, str, 10 );
            strcat ( aux, str );
            strcat ( aux, "_" );
            itoa ( ebus, str, 10 );
            strcat ( aux, str );
            idInst = _s2a ( aux );
            KpcMakeInstance ( idInst, _a2o ( idObj ) );
            bus_number ( ibus, aux );
            KpcSetValueF ( _a2o ( idInst ), _s2a ( "InitBus"
), _s2a ( aux ) );
            bus_number ( ebus, aux );
            KpcSetValueF ( _a2o ( idInst ), _s2a (
"EndBus" ), _s2a ( aux ) );
            KpcSetValueF ( _a2o ( idInst ), _s2a (
"Circuits" ), _i2a ( ncir ) );
            KpcSetValueF ( _a2o ( idInst ), _s2a (
"FlowLimit" ), _f2a ( flow_lim ) );
            j++;
        };
    } while ( !strchr ( line, '*' ) );
    if ( type == 1 ) j = 1;
    fsetpos ( fp, &filepos );
}

float extract ( char source[72], int begin, int end )
{
    char buffer[12];
    int i;
    float num;

    for ( i=begin; i<end; i++) buffer[i-begin] = source[i];
    buffer[i-begin] = '\0';
    num = atof (buffer);
    return num;
}

void bus_number ( int bus, char dest[11] )
{
    char str[4];
    strcpy ( dest, "Bus_" );
    itoa ( bus, str, 10 );
    strcat ( dest, str );
}
```

b) Segunda DLL - leitura e atualização dos dados de mercado (geração e carga) para o ano corrente.

Esta DLL é responsável por ler, no arquivo do caso base, os valores de geração e carga para todas os barramentos do sistema no ano corrente. O ano corrente está presente na base de conhecimentos e foi definido pelo usuário.

```
/*
    DLL SapYearData
    Autor: Hercídio de Paula Silveira Neto
    (Eng. Eletricista)

    Parâmetros de Chamada:
    1) Nome do Arquivo do Caso Base

    Atualizar Base de Fatos no KAPPA-PC
        - Bus > dados de barras;
            . Leitura de Dados de Geração para um o ano
presente.
            . Leitura de Dados de Carga para um o ano
presente.

    Lista com rotinas Kpc e Kpa utilizadas:
    . KpaArgCount      / KpaReturnError /
KpaGetArgAtom
    . KpaReturnTrue /
    . KpcGetValue / KpcSetValueF
    . _a2o, _i2a, f2a, _s2a, _u2i
*/

# include "kappadll.h"
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# include <ctype.h>
# include <conio.h>

void readata ( FILE *fp );

void study_slots ( FILE *fp, int type );

float extract ( char source[80], int begin, int end );

short PEXPORT SapYearData ( ARGLIST lpArgList )
{
    FILE *fp1;
    ATOMID idAtom;
    char file_base[40];

    if ( KpaArgCount ( lpArgList ) != 1 )
        KpaReturnError;
    if ( !KpaGetArgAtom ( lpArgList, 1, idAtom ) )
        KpaReturnError;
    _a2s ( file_base, idAtom );
    if (( fp1 = fopen ( file_base, "rt" )) == NULL)
        KpaReturnError;
    readata ( fp1 );
    fclose ( fp1 );
    KpaReturnTrue;
}

void readata ( FILE *fp )
{
```

```
    char line[72];
    char ch;

    while ( !feof ( fp ))
    {
        ch = fgetc ( fp );
        if ( ch == '*' )
        {
            fgets ( line, 10, fp );
            if ( !strchr ( line, '*' ) )
            {
                strlwr ( line );
                line[0] = _toupper ( line[0] );
                if ( strstr ( line, "Gat" ) )
                    study_slots ( fp, 1 );
                if ( strstr ( line, "Cat" ) )
                    study_slots ( fp, 2 );
            };
        };
    };
}

void study_slots ( FILE *fp, int type )
{
    ATOMID idObj, idSlot, idInst;
    fpos_t filepos;
    char line[80], str[4], aux[11];
    int year, ny, num, i, k=1;
    float value;

    idObj = _s2a ( "Study" );
    _a2s ( str, KpcGetValueF ( _a2o ( idObj ), _s2a (
"Present" )));
    year = atoi ( str );
    idSlot = _s2a ( "Power" );
    do {
        fgets ( line, 80, fp );
    } while ( strchr ( line, ';' ) );
    ny = extract ( line, 0, 2 );
    for ( i= 0; i<ny; i++)
    {
        num = extract ( line, k+2, k+4 );
        if ( num == year ) break;
        k+=3;
    };
    do {
        fgetpos ( fp, &filepos );
        fgets ( line, 80, fp );
        if ( !strchr ( line, ';' ) && !strchr ( line, '*' )
            && !strchr ( line, '#' )) {
            num = extract ( line, 0, 4 );
            value = extract ( line, 14+8*i, 21+8*i );
            strcpy ( aux, "Gen_");
            if ( type == 2 ) strcpy ( aux, "Load_" );
            itoa ( num, str, 10 );
            strcat ( aux, str );
            idInst = _s2a ( aux );
            KpcSetValueF ( _a2o ( idInst ), idSlot, _f2a (
value ));
        };
    } while ( !strchr ( line, '*' ) && !strchr ( line, '#' ));
    fsetpos ( fp, &filepos );
}

float extract ( char source[80], int begin, int end )
{
    char buffer[12];
    int i;
    float num;

    for ( i=begin; i<end; i++) buffer[i-begin] = source[i];
    buffer[i-begin] = '\0';
    num = atof (buffer);
    return num;
}
```

c) Terceira DLL - leitura e atualização do fluxo de potência nos ramos de transmissão da rede em estudo.

Esta DLL é responsável por manter atualizado o valor do carregamento de potência ativa nos ramos de transmissão, através da leitura do arquivo de resultados gerado pelo motor de simulação do protótipo SAPIENS.

```
/*
    DLL SapReadFlow - Leitura do Arquivo com Fluxo de
Potencia
    Autor: Hercídio de Paula Silveira Neto (Eng.
Eletricista)

    Utiliza um arquivo:
    1) Arquivo de resultados do programa sintra2.exe
(versão Hercídio).
    Base de Conhecimentos no KAPPA-PC
        - Bus > dados de barras;
        - Base > dados de Trafos (1) e LT's (2) existentes;
        - Planned > dados de circuitos a adicionar;
        - Expanded > dados dos circuitos adicionados na
expansao;
        - Study > lista com anos de estudo.

    Lista com rotinas Kpc e Kpa utilizadas:
    . KpaArgCount      / KpaReturnError /
KpaGetArgAtom
    . KpaReturnTrue / KpcMakeInstance / KpcSetValueF
    . _a2o, _i2a, f2a, _s2a*/

# include "kappadll.h"
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
```

```
# include <ctype.h>
# include <conio.h>

void readata ( FILE *fp);

void read_flow ( FILE *fp);

float extract ( char source[72], int begin, int end );

short PEXPORT SapReadFlow ( ARGLIST lpArgList )
{
    FILE *fp;
    ATOMID idAtom;
    char file_res[40], str[72];

    if ( KpaArgCount ( lpArgList ) != 1 )
        KpaReturnError;
    if ( !KpaGetArgAtom ( lpArgList, 1, idAtom ) )
        KpaReturnError;
    _a2s ( file_res, idAtom );
    if (( fp = fopen ( file_res, "rt" )) == NULL)
        KpaReturnError;
    readata ( fp);
    KpaReturnTrue;
}

void readata ( FILE *fp )
{
    char line[82];
    char ch;

    while ( !feof ( fp ))
    {
        fgets ( line, 82, fp );
        if ( strstr ( line, "COM : LISTA FLUXO ///" ))
        {
            read_flow ( fp );
            break;
        };
    };
}

void read_flow ( FILE *fp)
{
    ATOMID idInst;
    char line[82], tipo[2], aux[11], str[4];
    fpos_t filepos;
    int ibus=0, ebus=0, i=0;
    float flow;

    do
    {
        fgets ( line, 82, fp );
        if ( !strstr ( line, "COM : FIM" ))
        {
            ibus = extract ( line, 0, 5 );
            ebus = extract ( line, 31, 35 );
            flow = extract ( line, 61, 67 );
            tipo[0]=line[49];
            tipo[1]=line[50];
            if ( ibus > 0 ) i = ibus;
            if ( ebus > 0 )
            {
                strcpy ( aux, "Tr_");
                if ( strstr ( tipo, "LT" )) strcpy ( aux,
"Line_");
                itoa ( i, str, 10 );
                strcat ( aux, str );
                strcat ( aux, "_" );
                itoa ( ebus, str, 10 );
                strcat ( aux, str );
                idInst = _s2a ( aux );
                if (KpcInstance_QSM (idInst))
                    KpcSetValueF(
_a2o(idInst),_s2a("Flow"), _f2a(flow));
                strcpy ( aux, "TrP_");
                if ( strstr ( tipo, "LT" )) strcpy ( aux,
"LineP_");
                itoa ( i, str, 10 );
                strcat ( aux, str );
                strcat ( aux, "_" );
                itoa ( ebus, str, 10 );
                strcat ( aux, str );
                idInst = _s2a ( aux );
                if (KpcInstance_QSM (idInst))
                    KpcSetValueF(
_a2o(idInst),_s2a("Flow"), _f2a(flow));
            };
        };
    } while (( !strstr ( line, "COM : FIM")) && ( !feof ( fp)));
    fclose ( fp );
}

float extract ( char source[72], int begin, int end )
{
    char buffer[12];
    int i;
    float num;

    for ( i=begin; i<end; i++) buffer[i-begin] = source[i];
    buffer[i-begin] = '\0';
    num = atof (buffer);
    return num;
}
```

Com esta exposição, ficam descritos como se apresentam internamente as DLL de uso do protótipo SAPIENS.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BALDWIN, C.J. - DeSALVO, C.A. - HOFFMAN, C.H. et al. *A Model for Transmission Planning by Logic.* AIEE System Engineering Commitee. AIEE Fall General Meeting, Chicago, Ill., Oct. 11-16, 1959. pp. 1638-1645 - Feb. 1960

[2] BALU, N.J. *Section 4 - Present Expansion Modeling Techniques.* EPRI - Power Systems Planning and Operation Program - Electrical Systems Division. Long Range Transmission Expansion Models. TPS 79-728 Final Report. Oct. 1980. 27 p.

[3] BALU, N.J. *Section 5 - Commercially Available Programs.* EPRI - Power Systems Planning and Operation Program - Electrical Systems Division. Long Range Transmission Expansion Models. TPS 79-728 Final Report. Oct. 1980. 5 p.

[4] BALU, Neal J., ADAPA, Rambabu, CAULEY, Gerry et al. *IEEE Review of Expert Systems in Bulk Power System Planning and Operation.* Proceedings of the IEEE, Vol. 80, No. 5, p. 727-731, May 1992;

[5] BERRY, J.T. *Programando em C++.* Makron, McGraw-Hill 1991;

[6] BINATO, Silvio, OLIVEIRA, Gerson C., SOUTO, Ronaldo V. *Programa PLANEJA - Manual do Usuário.* CEPEL. Relatório Técnico 025C/81, 69 p., 22 Jan. 1981.

[7] BINATO, Silvio, OLIVEIRA, Gerson C., SOUTO, Ronaldo V. *Programa SINTRA - Manual do Usuário.* CEPEL Relatório Técnico 276/92, 83 p., 11 Ago. 1992.

[8] BINATO, Silvio., OLIVEIRA, Gerson C. *Planejamento Dinâmico Aplicado à Expansão de Redes de Transmissão.* ACSI-CEPEL. 7 p. CBA 1992.

[9] BRAUNER, Günter, ZOBEL, Manfred. *Knowledge Based Planning of Distribution Networks.* IEEE Transactions on Power Systems, Vol. 9, No. 2, May 1994.

[10] CUNHA, Sérgio H.F. *Desenvolvimento de Programas para o Planejamento da Expansão de Redes de Transmissão.* ELETROBRÁS - DEST. 2 Fev. 1989. 8 p. ECE-259/78.

[11] CUNHA, Sérgio H.F. *Expansão e Desenvolvimento de Programas de Síntese de Redes de Transmissão.* ELETROBRÁS - DEST. 1978, 6 p. PTR-1 259/78.

[12] *Dados Básicos para Estudos de Planejamento da Transmissão a Longo Prazo.* ELETROSUL - Comitê de Planejamento. Nota Técnica NTC-002/86. 55 p. 23 Abr. 1986.

[13] DAVID, A.K., RONG-DA, Zhao. *Integrating Expert Systems With Dynamic Programming in Generation Expansion Planning.* IEEE Transactions on Power Systems, Vol. 4, No. 3, p. 1095-1101, Aug. 1989.

[14] DILLON, Tharam S., LAUGHTON, Michael A. *Expert System Applications in Power Systems.* Prentice Hall International Series in Power Systems Computation. 1990, 408 p.

[15] DOULLIEZ, P. *Optimal Capacity Planning of Multidimensional Networks.* Tese Doutorado, Universitè Catholique de Louvain, Bélgica, 1970.

[16] ELETROBRÁS - EMPRESAS - CEPEL. *Resumo das Diretrizes da ELETROBRÁS sobre o Desenvolvimento de Programas de Síntese de Transmissão (SINTRA e Programas Correlatos).* 7 p. 7 Jul. 1987.

[17] ELETROBRÁS. GCPS. CTST. GTCP. *Metodologias para Comparação Econômica de Alternativas de Plano de Expansão e Análise de Viabilidade com Enfoque Social de Projetos de Transmissão.* 62 p. 21 Dez. 1987.

[18] ELGERD, O.I. *Introdução à Teoria de Sistemas de Energia Elétrica.* McGraw-Hill. 1976.

[19] *F77L-EM/32 - Fortran Language System Reference Manual.* Lahey Computer Systems, Inc. Revision B, 503 p., June 1989.

[20] FELDMAN, S.I., GAY, David M., MAIMONE, Mark W. et al. *A Fortran to C Converter.* AT&T Bell Laboratories. Computing Science Technical Report No. 149. May 16, 1990. 25 p.

[21] FOLEY, Mike, BOSE, Anjan, MITCHELL, William et al. *An Object Based Graphical User Interface for Power Systems.* IEEE Transactions on Power Systems, Vol. 8, No. 1, p. 97-104, Feb. 1993.

[22] FONSECA, Luiz G.S., LÚCIO, João C.M. *Planejamento da Expansão do Sistema de Geração considerando Redes de Transmissão.* UFSC-EEL-GSP-GRUDI. 8º CBA - UFPa - Belém-PA, 1990. p. 838-843.

[23] FONSECA, Luiz.G.S., TEIVE, Raimundo C.G. *Aquisição de Conhecimento e Desenvolvimento de um Sistema Especialista para o Planejamento Estático da Transmissão.* UFSC - Laboratório de Planejamento (LABPLAN). Depto. Eng. Elétrica. 6 p. 1993.

[24] FORTUNATO, Luiz Alberto Machado, ARARIPE Neto, Tristão de Alencar et al. *Introdução ao Planejamento da Expansão e Operação de Sistemas de Produção de Energia Elétrica.* Niterói: Universidade Federal Fluminense, EDUFF, 1990. 232 p.

[25] GALIANA, Francisco D., McGILLIS, Donald T., MARIN, Miguel A. *Expert Systems in Transmission Planning.* Proceedings of the IEEE, Vol. 80, No. 5, p. 712-724, May 1992,;

[26] GARVER, L.L. *Experiences with Applications of a Transmission Planning Program.* GE, General Electric Co. Electric Utility Systems Engineering Department. EPRI Transmission Planning Workshop, New York-NY. June 2-3, 1980. Section 9, 15 p.

[27] GLOVER, J.D., DOW, Leonard F. *Student Design Projects in Power Engineering.* IEEE Transactions on Power Systems, Vol. 5 No. 4, p. 1390-1398. Nov. 1990.

[28] GONZAGA, Clóvis C., PERSIANO, Ronaldo C.M. *Otimização e Expansão de Sistemas de Transmissão de Energia.* COPPETEC - COPPE/UFRJ. Rio de Janeiro, 1979. 103 p. Relatório Final ET-2130/79.

[29] HARMON, P., KING, D. *Expert Systems - Artificial Intelligence in Business.* John Wiley & Sons, New York-NY, 1985

[30] HOLEN, Arne T., BOTNEN, Åsmund, STØA, Peter et al. *Coupling Between Knowledge-Based and Algorithmic Methods.* Proceedings of the IEEE, Vol.80, No. 5, p. 745-756, May 1992.

[31] HWA, Lee K. *Planejamento da Expansão a Longo Prazo de Redes de Transmissão de Energia Elétrica usando Técnicas de Sistemas Baseados em Conhecimentos.* Dissertação de Mestrado em Engenharia de Sistemas e Computação, COPPE/UFRJ, Jul. 1987.

[32] KALTENBACH, S.C. *Problems Relating to the Optimal Planning of Network Capacities.* Tese de Doutorado, Dept. of Industrial Engineering Stanford University, Stanford, 1969.

[33] *KAPPA - PC Advanced Topics, Version 2.0.* Intelicorp Inc. June 1992. 317 p. Il.

[34] *KAPPA - PC Reference Manual, Version 2.0.* Intelicorp Inc. June 1992. 291 p. Il.

[35] *KAPPA - PC User's Guide, Version 2.0.* Intelicorp Inc. June 1992. 371 p. Il.

[36] LEE, S., HIDES, K. e HMYILICZA, E. *Transmission Expansion by Branch and Bound Integer Programming with Optimal Cost-Capacity Curves.* IEEE Transactions on Power Apparatus Systems. PAS-93, p. 1390, Set.-Out. 1974.

[37] LI, S., SHAHIDEHPOUR, S.M., WANG, C. *Promoting the Application of Expert Systems in Short-Term Unit Commitment.* IEEE Transactions on Applied Superconductivity, Vol. 3, No. 1, p. 286-292. Mar. 1993.

[38] LIN, C.E., HUANG, C.J., HUANG, C.L. et al. *An Expert System for Generator Maintenance Scheduling Using Operation Index.* IEEE Transactions on Power Systems. p. 1141-1148. 1992.

[39] LIU, Chen-Ching, LIOU, Kan-Lee, CHU, Ron F. et al. *Generation Capability Dispatch for Bulk Power System Restoration: A Knowledge Aproach.* IEEE Transactions on Power Systems, Vol. 8, No. 1, p. 316-325. Feb. 1993.

[40] LIU, C.C. et al. *Practical use of Expert Systems in Planning and Operation of Power Systems.* CIGRÉ Task Force 38.06.03. ÉLECTRA No. 146, p. 31-67, Feb. 1993.

[41] LIU, Cheng C., SON, Paul V. *Expert Systems Development Experience and User Requirements.* CIGRÉ Task Force 38.06.03. ÉLECTRA No. 146, p. 29, Feb. 1993.

[42] LÚCIO, João C.M. *Planejamento da Expansão do Sistema de Geração considerando Redes de Transmissão.* Dissertação de Mestrado em Engenharia Elétrica, UFSC, Dez. 1990.

[43] MANZONI, G., PARIS, L., VALTORTA, M. *Power Systems Planning Practice in Italy.* IEEE Transactions on Power Apparatus and Systems, Vol. PAS-98, No. 3, pp. 689-699, May/June 1979.

[44] MINUSSI, Carlos R. *Controle de Segurança Dinâmica em Sistemas de Energia Elétrica.* Tese de Doutorado em Engenharia Elétrica. UFSC, Nov./1990.

[45] MONTICELLI, A. *Métodos de Análise e Síntese aplicados ao Planejamento a Longo Prazo de Sistemas de Transmissão de Energia Elétrica.* UNICAMP - Depto. Eng. Elétrica da Faculdade de Engenharia de Campinas. Campinas-SP, 1979. 189 p.

[46] MONTICELLI, A. *Planejamento de Redes de Transmissão de Energia Elétrica.* Universidade Estadual de Campinas, Depto. Eng. Elétrica da Faculdade de Engenharia de Campinas. Relatório nº2 do Convênio CEPEL / UNICAMP sobre Planejamento de Redes de Transmissão, Campinas 1980. 51 p.

[47] MONTICELLI, A., CUNHA, S.H. et al. *Interactive Transmission Network Planning Using a Least-Effort Criterion.* IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 10, p. 3919-3925. Oct. 1982.

[48] MONTICELLI, Alcir (UNICAMP); PEREIRA, Mário V. F. (CEPEL) et al. *Planejamento Interativo de Redes de Transmissão considerando Contingências: O Sistema SINTRA.* VII Seminário Nacional de Produção Transmissão de Energia Elétrica. GRUPO VII, Planejamento de Sistemas Elétricos (GPL). 20 p. BSB/GPL/01. Brasília, 1984.

[49] MOROZOWSKI, Marciano. *Matrizes Esparsas em Redes de Potência - Técnicas de Operação.* Livros Técnicos e Científicos Editora, Rio de Janeiro 1981.

[50] MOROZOWSKI, Marciano, FONSECA, Luiz G.S. *Aplicação do Programa SINTRA no Planejamento de Redes de Transmissão de Energia Elétrica.* ELETROSUL. DVPT. 21 Jan. 1983. 32 p. Nota Técnica NTC 002/83

[51] MOROZOWSKI, Marciano, FONSECA, Luiz G.S., CRUZ, Carlos R.G. *Uso de Modelos de Síntese no Planejamento de Sistemas de Transmissão: A Experiência da ELETROSUL.* VII Seminário Nacional de Produção Transmissão de Energia Elétrica. GRUPO VII Planejamento de Sistemas Elétricos (GPL). Brasília-DF, 1984. 14 p. BSB/GPL/10.

[52] MOROZOWSKI, Marciano. *Notas de Aula.* Universidade Federal de Santa Catarina, Centro Tecnológico, Grupo de Planejamento de Sistemas de Energia Elétrica. 100 p., 1994.

[53] NEYER, Andreas F., WU, Felix F., IMHOF, Karl. *Object-Oriented Programming for Flexible Software Example of a Load Flow.* IEEE Transactions on Power Systems, Vol. 5, No. 3, p. 689-696, Aug. 1990.

[54] NILSSON, Nils J. *Problem - Solving Methods in Artificial Intelligence.* 255 p. 1971.

[55] *OBJECTWINDOWS for C++, User's Guide.* Borland Inc. 393 p. 1992.

[56] OLIVEIRA, Gerson, PEREIRA, Mário V.F., SOUTO, Ronaldo V.S. et al. *Proposta de Programa de Trabalho para Conclusão do SINTRA em sua Versão Inicial para Divulgação junto às Empresas.* 5 p. 13 Jun. 1987.

[57] OLIVEIRA, G.C. *Síntese Interativa de Redes de Transmissão.* CEPEL. Rio de Janeiro, 1992.

[58] PINTO, Leontina M.V.G., SANTOS, Dorel S. *Planejamento da Expansão da Geração/Transmissão Sujeito a Restrições Financeiras.* 23 p. 1990.

[59] PUNTEL, W.R. et al. *An Automated Method for Long-Range Planning of Transmission Networks.* PICA Conference, Chicago, Abril, 1973

[60] RAHMAN, Saifur. *Artificial Intelligence in Electric Power Systems, A Survey of The Japanese Industry.* IEEE Transactions on Power Systems, Vol. 8, No. 3, p. 1211-1218. Aug. 1993.

[61] SANTOS, A.S.Jr., FRANÇA, P.M., SAID, A. *An Optimization Model for Long-Range Transmission Expansion Planning.* UNICAMP - Faculdade de Engenharia Elétrica. IEEE Transactions on Power Systems, Vol. 4, No. 1, p. 94-101. Feb., 1989

[62] SLYE, Fintan, O'MALLEY, Mark. *An Expert System for the Restoration of an Island Power System.* University College Dublin, Ireland, 1993. UPEC 93. p. 530-533.

[63] SON, Paul V. *Exploring User Requirements of Expert Systems in Power System Operation and Control.* CIGRÉ Task Force 39.03. ÉLECTRA No. 146, p. 69-83, Feb. 1993.

[64] SCHILDT, Herbert. *Turbo C++: Guia do Usuário.* Makron Books, McGraw-Hill, 1992.

[65] TELLO, E.R.T.. *Windows Programação Orientada para Objetos.* Tradução de Daniel Vieira. LTC - Livros Técnicos e Científicos. Rio de Janeiro, 1992.

[66] TINNEY, W.F. *Computer Aided Transmission Planning.* Bonneville Power Administration. 1975?

[67] TINNEY, William F. *Power Systems Planning and Operations; Future Problems and Research Needs.* EPRI Report. EL - 377 - SR - February 1977.

[68] TINNEY, William F. *Mathematical Challenges in Power System Planning.* Electric Power Problems, The Mathematical Challenge. Siam, Philadelphia, 1980. 17 p.

[69] *Turbo C++ Version 3.0 - User's Guide.* Borland International - 1992.

[70] TUTHILL, G. Steven. *Knowledge Engineering: concepts and practices for knowledge-basede systems.* 372 p. 1990.

[71] UNIVERSIDADE FEDERAL DE SANTA CATARINA - Centro Tecnológico - Departamento de Engenharia Elétrica - Curso de Pós-Graduação em Engenharia Elétrica. *Normas para Apresentação de Dissertações e Tese.*

[72] WATERMAN, D.A. *A Guide to Expert System.* Library of Congress Cataloging in Publication Data. 419 p. 1986.

[73] WATERMAN, Donald A., PHILIP, Klahr. *Expert Systems Techniques, Tools and Applications.* 441 p. 1986.

[74] WOLLENBERG, Bruce F., SAKAGUCHI, Toshiaki. *Artificial Intelligence in Power System Operations.* Proceedings of the IEEE, Vol. 75, No. 12, p. 1678-1685, Dec., 1987.

[75] WHYSONG, J.L., URAM, R., BROWN, H.E. et al. *Computer Program for Automatic Transmission Planning.* AIEE System Engineering Committee. AIEE Winter General Meeting. p. 774-781, Feb. 1963.